Foxit PDF SDK

开发者指南
FOXIT PDF SDK

TABLE OF CONTENTS

1 FOXIT PDF SDK 简介

您是否曾经想要构建一个可以对 PDF 文档进行任何操作的应用程序？如果您的答案是 "Yes", 那么恭喜您！您找到了业界中可以构建稳定、安全、高效且功能齐全的 PDF 应用的优选解决方案。

Foxit PDF SDK 提供高性能的开发库，帮助软件开发人员使用最流行的开发语言和环境在不同平台 (包括 Windows、Mac、Linux、Web、Android、iOS 和 UWP) 的企业版、移动版和云应用程序中添加强大的 PDF 功能。

1.1 为什么选择 Foxit PDF SDK

Foxit 是领先的 PDF 软件解决方案供应商，专注于 PDF 显示、编辑、创建、管理以及安全方面。Foxit PDF SDK 开发库已在当今许多知名的应用程序中使用，并且经过长期的测试证明 Foxit PDF SDK 的质量、性能和功能正是业界大部分应用程序所需要的。选择 Foxit PDF SDK 产品的几大理由：

- **易于集成**

开发人员可以将 SDK 无缝集成到他们自己的应用程序中。

- **轻量级**

部署简单快速，占用系统资源少。

- **支持跨平台**

支持当前主流的平台，比如 Windows、Mac、Linux、Web、Android、iOS 和 UWP。

- **基于福昕高保真的 PDF 渲染引擎**

Foxit PDF SDK 的核心技术是基于世界众多知名企业所信赖的福昕 PDF 引擎。福昕强大的 PDF 引擎可快速解析和渲染文档，不受设备环境的约束。

- **优秀的技术支持**

福昕对自己的开发产品提供了优秀的技术支持，当您在开发关键重要的产品时，可以提供高效的帮助和支持。福昕拥有一支 PDF 行业优秀的技术支持工程师团队，同时将定期地进行版本更新发布，通过添加新的功能和增强已有的功能来提升用户体验。

1.2 Foxit PDF SDK for C++ API

使用 Foxit PDF SDK 的应用程序开发人员可以利用 Foxit 强大、符合标准的 PDF 技术来安全地显示、创建、编辑、注释、格式化、组织、打印、共享、保护、搜索文档，以及填写 PDF 表单。此外，Foxit PDF SDK (C++和.NET) 包含一个内置可嵌入的 PDF Viewer，使开发过程更容易和更快捷。有关更多详细信息，请访问网站 https://developers.foxitsoftware.cn/pdf-sdk/。

在本手册中，我们专注于介绍 Windows、Linux 和 Mac 平台的 Foxit PDF SDK for C++ API。

Foxit PDF SDK for C ++ API 提供简单易用的 API，帮助 C++开发人员将强大的 PDF 技术无缝集成到他们自己的 Windows、Linux 和 Mac 平台项目中。并且提供了 PDF 文档相关的丰富功能，比如 PDF

浏览、书签导航、文本选择/复制/搜索、PDF 签名、PDF 表单、权限管理、PDF 注释以及全文搜索等。

1.3 评估

用户可申请下载 Foxit PDF SDK 的试用版本进行试用评估。试用版除了有试用期 10 天时间的限制以及生成的 PDF 页面上会有试用水印以外，其他都和标准认证版一样。当试用期到期后，用户需联系福昕销售团队并购买 licenses 以便继续使用 Foxit PDF SDK.

1.4 授权

程序开发人员需购买 licenses 授权才能在其解决方案中使用 Foxit PDF SDK。Licenses 授予用户发布基于 Foxit PDF SDK 开发的应用程序的权限。然而，在未经福昕软件公司授权下，用户不能将 Foxit PDF SDK 包中的任何文档、示例代码以及源代码分发给任何第三方机构。

1.5 关于此文档

此文档适用于需要使用 C++开发语言将 Foxit PDF SDK 集成到自己的应用程序中的开发人员。它旨在介绍 SDK 包结构和 SDK 的用法。

2 入门指南

安装并集成 Foxit PDF SDK 非常简单。本手册将提供 SDK 包的简要介绍。作为跨平台产品，Foxit PDF SDK 支持 Windows、Linux 和 Mac 桌面平台统一的接口。本章的主要内容是介绍系统要求、SDK 包结构、以及如何运行 demo 和创建自己的项目。

2.1 系统要求

平台	系统要求	备注
Windows	Windows Vista, 7, 8, 10 (32-bit 和 64-bit) Windows Server 2003, 2008, 2012 (32-bit 和 64-bit)	仅支持 Windows 8/10 经典样式，不支持 Store APP 和 Universal App。
Linux	x86/x64 (32-bit 和 64-bit OS) armv7/armv8 从 8.4 版本开始，Foxit PDF SDK for Linux（x86 和 x64）支持的最低 GCC 编译器版本已从 gcc4.8 升级到 gcc4.9.4。	所有的 Linux (x86/x64) 示例在 Ubuntu14.0 32/64 bit 上进行过测试。 所有的 Linux (armv7/armv8) 示例都在 armv7 或者 armv8 系统中进行过测试。
Mac	Mac OS X 10.6 及以上 (x64) Mac OS 12.0 及以上 (arm64)	

2.2 Windows

在本手册中，请知晓：图片上面高亮的矩形区域指的是 SDK 的版本号，当前 SDK 的版本是 10.0，则其代表 10_0。

2.2.1 包结构说明

下载 Foxit PDF SDK for Windows 包，解压到一个新的目录如 "foxitpdfsdk_10_0_win"，如 Figure 2-1 所示。其中解压包中包括如下的内容：

doc: API 手册，开发者指南

examples: 示例工程和 demo

include: Foxit PDF SDK API 的头文件

lib: SDK 库和授权文件

res: 输出预览 (output preview) demo 使用的默认 icc profile 文件

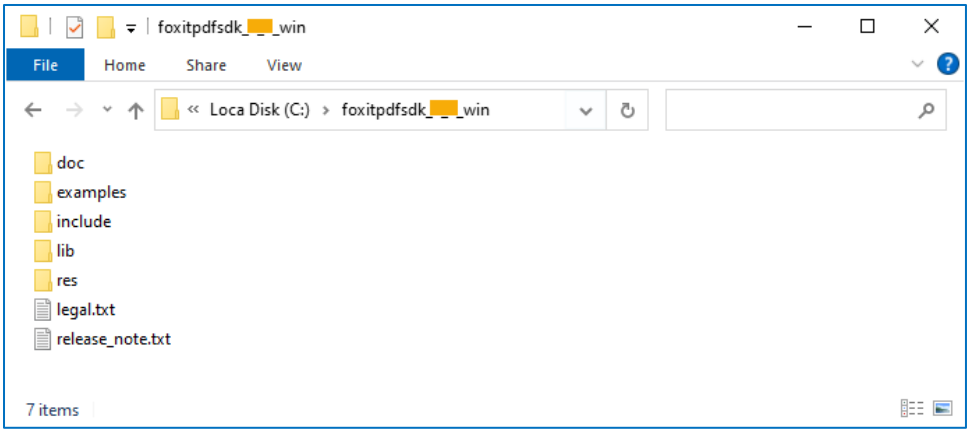


Figure 2-1

在 "examples" 文件夹下，有两种类型的 demo。"\examples\simple_demo" 包含 30 多个简单示例 demo，涵盖各种 PDF 应用程序。"\examples\view_demo" 包含一个 UI demo，该 demo 实现的是一个精简的 PDF 阅读器。

2.2.2 运行 demo

Simple Demo

Simple demo 工程提供了一些示例，向开发人员展示如何有效地应用 Foxit PDF SDK APIs 来完成其应用程序中所需的功能。

在 Visual Studio 中运行 demo (除了 security, signature, ocr, compliance, html2pdf, office2pdf, output preview, pdf2office, dwg2pdf 和 ofd demo，这些将在后面介绍)，您可以按照如下的步骤：

1) 在 "\examples\simple_demo" 文件夹下，根据您 Visual Studio 的版本双击 "simple_demo_vs2010.sln" 或者 "simple_demo_vs2015.sln" 或者 "simple_demo_vs2017.sln"或者 "simple_demo_vs2019.sln"或者 "simple_demo_vs2022.sln"。

2) 单击 "Build > Build Solution" 编译所有的 demo。或者，如果您只想编译某个特定的 demo，您可以右击该 demo 工程，然后选择 "Build"，或者在该 demo 工程的文件夹下双击 "*.vcxproj"，然后编译该工程。

   编译后，在 "\examples\simple_demo\bin" 文件夹下将生成 ".exe" 可执行文件。可执行文件的名称取决于工程编译的配置。

3) 运行某个特定的可执行文件，只需要双击即可。

   对于会生成输出文件 (pdf, 文本或者图片文件) 的 demo，会在 "\examples\simple_demo\output_files\" 文件夹下生成以该 demo 名称命名的文件夹，并且输出文件将会在该文件夹下生成。

***备注：**如果您需要查看详细的执行过程，您可以在命令行中运行。启动“cmd.exe”，导航到 "\examples\simple_demo\bin"，然后运行特定的可执行文件。*

Security demo

在运行 **security** demo 之前，您需要安装 "\examples\simple_demo\input_files" 文件夹下的 "foxit.cer" 和 "foxit_all.pfx" 证书。

a) 安装 "foxit.cer"，双击其启动证书导入向导。然后选择"Install certificate... > Next > Next > Finish"。

b) 安装 "foxit_all.pfx"，双击其启动证书导入向导。然后选择 "Next > Next > (在文本框中输入私钥的密码 "**123456**") ，然后点击 Next > Next > Finish"。

c) 参考其他 demo 的运行步骤运行该 demo。

Signature demo

在运行 **signature** demo 之前，您需要确保已经安装了 OpenSSL。从 OpenSSL 官网下载 OpenSSL 源码包，或者您可以直接与我们客服联系。获取到源码包后，解压并进行如下操作:

1) 将 OpenSSL 文件夹拷贝到 "include"文件夹下，以确保 demo 中引用的 OpenSSL 头文件可以被识别到。
2) 将 "libeay32.lib" 库拷贝到 "lib" 文件夹下。
3) 参考其他 demo 的运行步骤运行该 demo。

***备注：**OpenSSL 1.1.1-stable 版本在 signature demo 中已经验证是可用的。您可以替换为其他所需的版本，但可能需要做一些相应的更改。*

OCR 和 Compliance demo

对于 **ocr** 和 **compliance** demo，您需要首先构建一个资源目录，请联系 Foxit 支持团队或者销售团队获取相应的资源包。关于如何运行该 demo 的更详细的信息，请参考 3.35 小节 "OCR" 和 3.36 小节 "Compliance"。

HTML to PDF demo

对于 **html2pdf** demo，您需要首先联系 Foxit 支持团队或者销售团队获取 HTML 转 PDF 的引擎包。关于如何运行该 demo 的更详细的信息，请参考 3.38 小节 "HTML 转 PDF"。

Office to PDF demo

对于 **office2pdf demo**，请参考 3.39 小节 "Office 转 PDF"。

Output Preview demo

对于 **output preview demo**，您需要设置包含默认 icc profile 文件的文件夹路径。关于如何运行该 demo 的更详细信息，请参考 3.40 小节 "输出预览 (Output Preview)"。

PDF to Office demo

对于 **pdf2office** demo，您需要首先联系 Foxit 支持团队或者销售团队获取 PDF 转 office 的引擎包。关于如何运行该 demo 的更详细的信息，请参考 3.45 小节 "PDF 转 Office"。

Dwg to PDF demo

对于 **dwg2pdf** demo，您需要首先联系 Foxit 支持团队或者销售团队获取 DWG 转 PDF 的引擎包。关于如何运行该 demo 的更详细的信息，请参考 3.46 小节 "DWG 转 PDF"。

OFD demo

对于 **ofd** demo，您需要首先联系 Foxit 支持团队或者销售团队获取 OFD 的引擎包。关于如何运行该 demo 的更详细的信息，请参考 3.47 小节 "OFD"。

View Demo

View demo 为开发人员提供了一个使用 Foxit PDF SDK APIs 实现一个 PDF 阅读器的示例。

在 Visual Studio 中运行该 demo，根据您 Visual Studio 的版本双击 "\examples\view_demo\PDFReader\project" 文件夹下的 "PDFReader_VS2010.sln" 或者 "PDFReader_VS2015.sln" 或者 "PDFReader_VS2017.sln"或者 "PDFReader_VS2019.sln"或者 "PDFReader_VS2022.sln"，然后点击 "Debug > Start Without Debugging" 运行该 demo。当 demo 运行起来后，您将看到如下的文件选择对话框：

Figure 2-2

您可以将 PDF 文档拖动到上面的窗体 (Figure 2-2) 来打开该文档，然后通过向下滚动或者按住鼠标左键移动 PDF 页面来浏览文档内容。此外，您可以点击 "Annot > HighLight" 来高亮文本，如 Figure 2-3 所示。

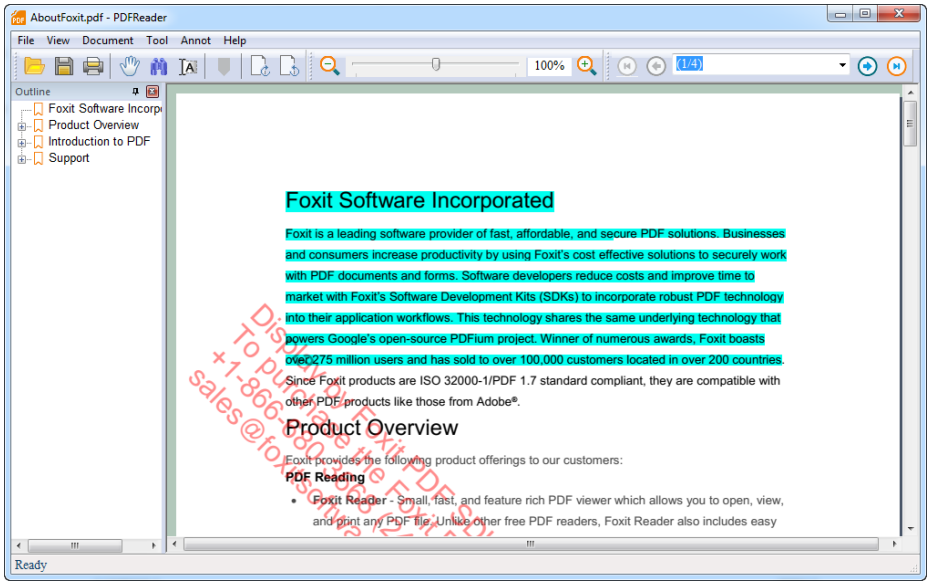


Figure 2-3

2.2.3 创建一个简单的工程

本节主要介绍如何使用 Foxit PDF SDK for Windows 创建一个简单的工程，该工程将 PDF 文档的首页渲染成 bitmap，然后将其另存为 JPG 图片。请按照如下的步骤操作：

1) 打开 Visual Studio，创建一个名为 "test_win" 的 Win32 控制台应用程序。

2) 将 "foxitpdfsdk_10_0_win" 文件夹下的 "include" 和 "lib" 文件夹拷贝到 "test_win" 工程目录下。

3) 添加 "include" 文件夹到 "Additional Include Directories"。在 Solution Explorer 中右击 *test_win* 工程，选择 "Properties"，然后找到 "Configuration Properties > C/C++ > General > Additional Include Directories"。

4) 在 test_win.cpp 的开头添加 include 头文件声明。

```
#include <iostream>
#include "../include/common/fs_common.h"
#include "../include/pdf/fs_pdfdoc.h"
#include "../include/pdf/fs_pdfpage.h"
#include "../include/common/fs_render.h"
```

5) 添加 using namespace 命名空间声明。

```
using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;
```

6) 包含 Foxit PDF SDK 库。

```
#if defined (_WIN64) // windows 64bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#endif

#elif defined (_WIN32) // windows 32bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#endif
#endif
```

7) 初始化 Foxit PDF SDK 库。在调用任何 APIs 之前，应用程序必须使用 license 授权码初始化 Foxit PDF SDK 库。试用 license 文件在 "lib" 文件夹下。

```
const char* sn = " ";
const char* key = " ";
foxit::ErrorCode code = Library::Initialize(sn, key);
if (code != foxit::e_ErrSuccess) {
        return FALSE;
}
```

备注：*参数 "sn" 的值在 "**gsdk_sn.txt**" 中 ("SN=" 后面的字符串)，"key" 的值在 "**gsdk_key.txt**" 中 ("Sign=" 后面的字符串)。*

8) 加载一个 PDF 文档，然后解析该文档的首页。假设您已经在 "test_win\test_win" 文件夹下放入了一个 "Sample.pdf" 文件。

```
PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
```

9) 将页面渲染成 bitmap，然后将其另存为 JPG 图片。

```
int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);
// Render page.
Renderer render(bitmap, false);
render.StartRender(page, matrix, NULL);

// Add the bitmap to image and save the image.
Image img;
img.AddFrame(bitmap);
img.SaveAs("testpage.jpg");
```

10) 点击 "Build > Build Solution" 编译该工程。可执行文件 "test_win.exe" 将在 "test_win\Debug" 或者 "test_win\Release" 文件夹下生成，具体取决于编译配置。

11) 将 "lib" 文件夹下的 "fsdk_win32.dll" 或者 "fsdk_win64.dll" 拷贝到输出目录 ("test_win\Debug" 或者 "test_win\Release")。请确保 "fsdk_win**.dll" 架构需要与应用程序的 platform target (Win32 或者 Win64) 相匹配。

12) 选择如下其中一种方式运行该工程：

    i. 在 Visual Studio 中点击 "Debug > Start Without Debugging" 运行工程，然后在 "test_win\test_win" 文件夹下将会生成 "testpage.jpg"。

ii. 双击可执行文件 "test_win.exe" 运行工程。使用这种方式，您需要将 "Sample.pdf" 文档放在与 "test_win.exe" 相同的文件夹中，并且 "testpage.jpg" 也将在同一文件夹中生成。

"test_win.cpp" 的完整内容如下：

```
#include "stdafx.h"

#include <iostream>
#include "../include/common/fs_common.h"
#include "../include/pdf/fs_pdfdoc.h"
#include "../include/pdf/fs_pdfpage.h"
#include "../include/common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Include Foxit PDF SDK library.
#if defined (_WIN64) // windows 64bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#endif

#elif defined (_WIN32) // windows 32bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#endif
#endif

int _tmain(int argc, _TCHAR* argv[])

{
	// 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=").
	const char* sn = " ";
	const char* key = " ";
	foxit::ErrorCode code = Library::Initialize(sn, key);
	if (code != foxit::e_ErrSuccess) {
		return FALSE;
	}
```

```
        // Load a PDF document, and parse the first page of the document.
        PDFDoc doc("Sample.pdf");
        ErrorCode error_code = doc.Load();
        if (error_code!= foxit::e_ErrSuccess) return 0;
        PDFPage page = doc.GetPage(0);
        page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

        int width = static_cast<int>(page.GetWidth());
        int height = static_cast<int>(page.GetHeight());
        Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

        // Prepare a bitmap for rendering.
        Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
        bitmap.FillRect(0xFFFFFFFF, NULL);
        // Render page.
        Renderer render(bitmap, false);
        render.StartRender(page, matrix, NULL);

        // Add the bitmap to image and save the image.
        Image img;
        img.AddFrame(bitmap);
        img.SaveAs("testpage.jpg");
        return 0;
}
```

2.3 Linux

2.3.1 Linux (x86/x64)

在本手册中，请知晓：图片上面高亮的矩形区域指的是 SDK 的版本号，当前 SDK 的版本是 10.0，则其代表 10_0。

2.3.1.1 包结构说明

下载 Foxit PDF SDK for Linux (x86/x64)包，解压到一个新的目录如 "foxitpdfsdk_10_0_linux"，如 Figure 2-4 所示。其中解压包中包括如下的内容：

doc: API 手册，开发者指南

examples: 示例工程和 demo

include: Foxit PDF SDK API 的头文件

lib: SDK 库和授权文件

res: 输出预览 (output preview) demo 使用的默认 icc profile 文件

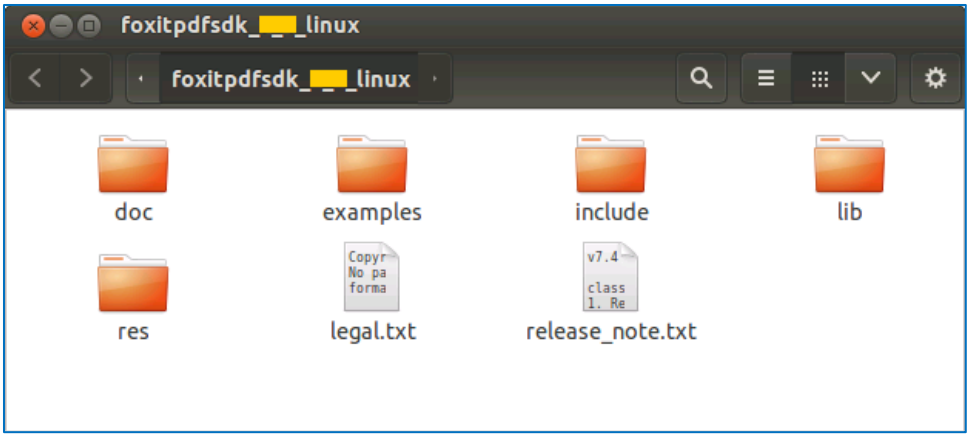


Figure 2-4

在 "examples" 文件夹下，有两种类型的 demo。"examples/simple_demo" 包含了大量的 PDF 应用程序的 demo。"examples/view_demo" 包含一个 Qt UI demo，该 demo 实现的是一个精简的 PDF 阅读器。

2.3.1.2 运行 demo

更新 GCC 编译器

从 8.4 版本开始，Foxit PDF SDK for Linux（x86 和 x64）支持的最低 GCC 编译器版本已从 gcc4.8 升级到 gcc4.9.4。为了确保 SDK 正常工作，请确保您当前的 GCC 版本为 4.9.4 或更高版本，或者 libstdc++.so.6 为 6.0.20 或更高版本。

Simple Demo

Simple demo 工程提供了一些示例，向开发人员展示如何有效地应用 Foxit PDF SDK APIs 来完成其应用程序中所需的功能。

在运行 demo 之前，请确保您已经正确配置环境，并且已安装 CMake 3.1 或更高版本。

OS	Tool chain	GLIBC
Linux x86/x64	gcc 4.9.4 或更高版本	GLIBC_2.17 或更高版本

在终端运行 demo (除了 security, signature, ocr (x64), compliance, htm2pdf, office2pdf, output preview, pdf2office, dwg2pdf and ofd (x64) demo，这些将在后面介绍)，您可以按照如下的步骤：

1) 打开一个终端，导航到 "foxitpdfsdk_10_0_linux/examples/simple_demo";
2) 运行 "**cmake -DPRJ_NAME=XXX**" 以编译指定的 demo。"XXX" 是 demo 的名称，该名称必须为 "annotation", "attachment", "pdf2text" 等等。例如，运行 "cmake -DPRJ_NAME=annotation"。
3) 运行 "**make**" 以构建上述命令中指定的 demo。然后将生成名为 "**XXX_xxx**" 的可执行文件。"XXX" 是 demo 的名称，"xxx" 是架构名，例如 "annotation_linux64"。

4) 执行 "**./XXX_xxx**" 以运行 demo。例如，执行 "./annotation_linux64" 来运行 annotation demo。

对于有输出文件 (pdf, 文本或者图片文件) 的 demo 来说，会在 "examples/simple_demo/output_files/" 文件夹下生成以该 demo 名称命名的文件夹，并且输出文件将会在该文件夹下生成。

Security 和 Signature demo

在运行 **security** 和 **signature** demo 之前，请确保您已经安装了 OpenSSL。从 OpenSSL 官网下载 OpenSSL 源码包，或者您可以直接与我们客服联系。获取到源码包后，解压并进行如下操作：

1) 将 OpenSSL 文件夹拷贝到 "include"文件夹下，以确保 demo 中引用的 OpenSSL 头文件可以被识别到。
2) 将 "libssl.a" 和 "libcrypto.a" 库拷贝到 "lib" 文件夹下。
3) 参考其他 demo 的运行步骤运行该 demo。

备注：*OpenSSL 1.0.2 版本在 security 和 signature demo 中已经验证是可用的。您可以替换为其他所需的版本，但可能需要做一些相应的更改。*

OCR

对于如何运行 **ocr** demo，请参阅 3.34 小节 "OCR"。

Compliance demo

对于如何运行 **compliance** demo，请参阅 3.35 小节 "Compliance"。

HTML to PDF demo

对于如何运行 **html2pdf** demo，请参阅 3.37 小节 "HTML 转 PDF"。

Office to PDF demo

对于 **office2pdf demo**，请参考 3.39 小节 "Office 转 PDF"。

Output Preview demo

对于如何运行 **output preview** demo，请参考 3.39 小节 "输出预览 (Output Preview)"。

PDF to Office demo

对于如何运行 **pdf2office** demo，请参考 3.45 小节 "PDF 转 Office"。

Dwg to PDF demo

对于如何运行 **dwg2pdf** demo，请参考 3.46 小节 "DWG 转 PDF"。

OFD demo (x64)

对于如何运行 **ofd** demo，请参考 3.47 小节 "OFD"。

View Demo

View demo 为开发人员提供了一个 Qt 示例，用来展示如何使用 Foxit PDF SDK APIs 实现一个 PDF 阅读器。

在运行 demo 之前，请确保您 Linux 系统已经安装了 compiler tool (GCC 4.8 或更高) 和 "Qt Creator" IDE。

在 Qt Creator 中运行 demo，请按照如下的步骤：

1) 打开 Qt Creator，点击 **Open Project**，然后导航到"examples/view_demo/PDFReader_Qt" 文件夹，选择 **PDFReader_Qt.pro** 文件打开该 demo。

   如果出现 "**Configure Project**" 窗口，请根据提示选择一种 kit, 然后点击 **Configure Project** 按钮，如图 Figure 2-5 所示。

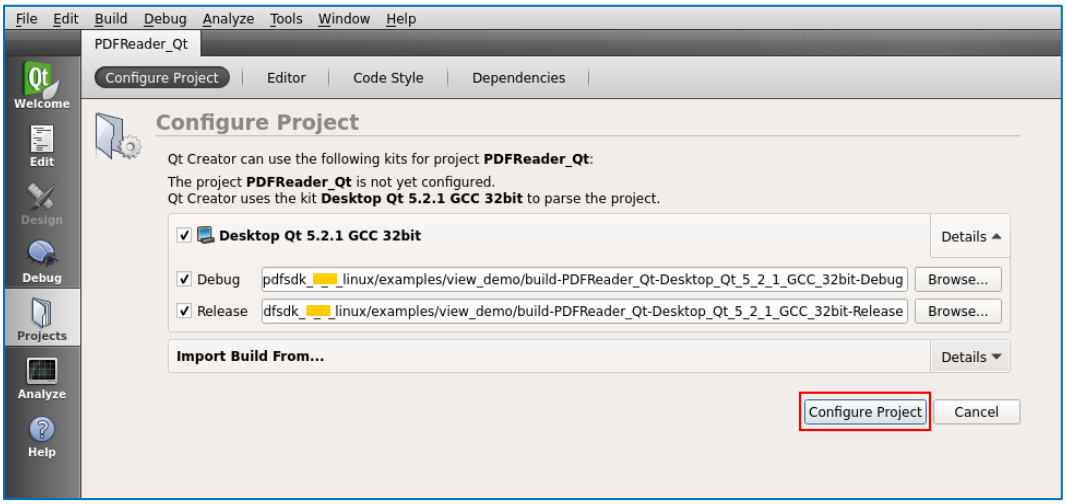


Figure 2-5

2) Demo 打开后，点击左侧工具栏的 **Projects**，将 "**Build&Run** -> **Build Settings** -> **General**" 下的 **Shadow build** 选项的勾选去掉，如图 Figure 2-6 所示。

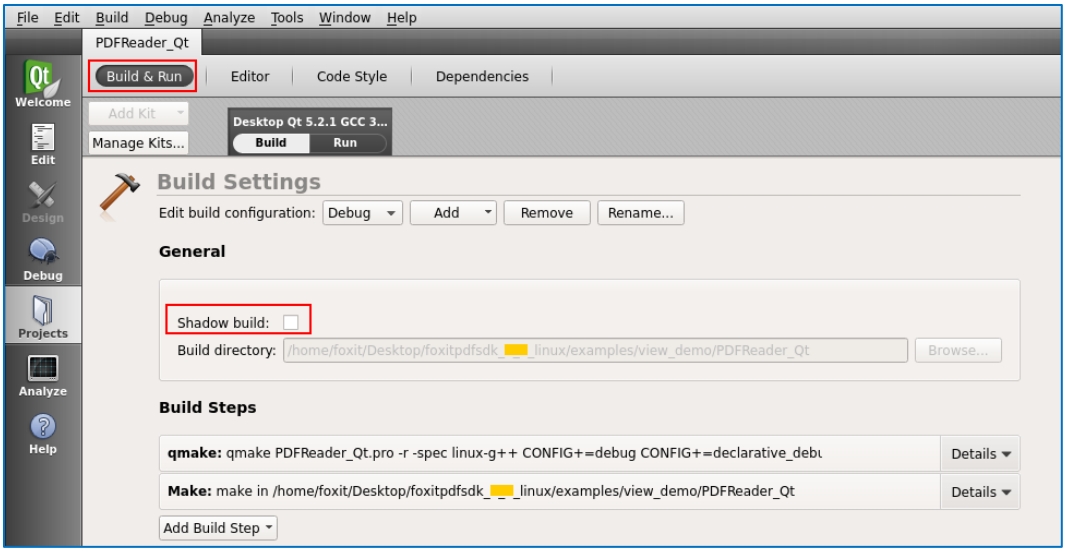


Figure 2-6

3) 编译并运行 demo。点击 **Build** -> **Run** 运行 demo。当 demo 运行起来后，您将看到如下的窗体：

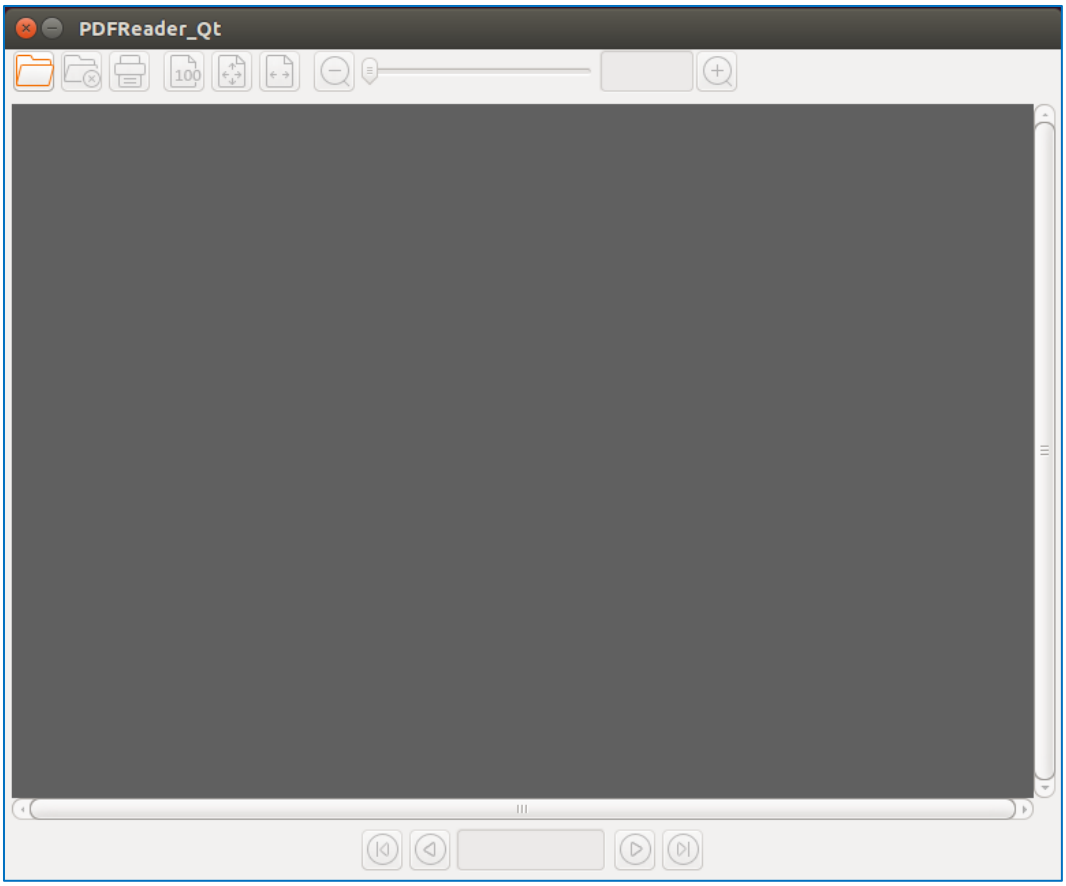


Figure 2-7

点击上面图片 (Figure 2-7) 的 按钮打开一个 PDF 文档。然后可以通过向下滚动来浏览文档内容，翻页，以及放大缩小 PDF 页面，如 Figure 2-8 所示。

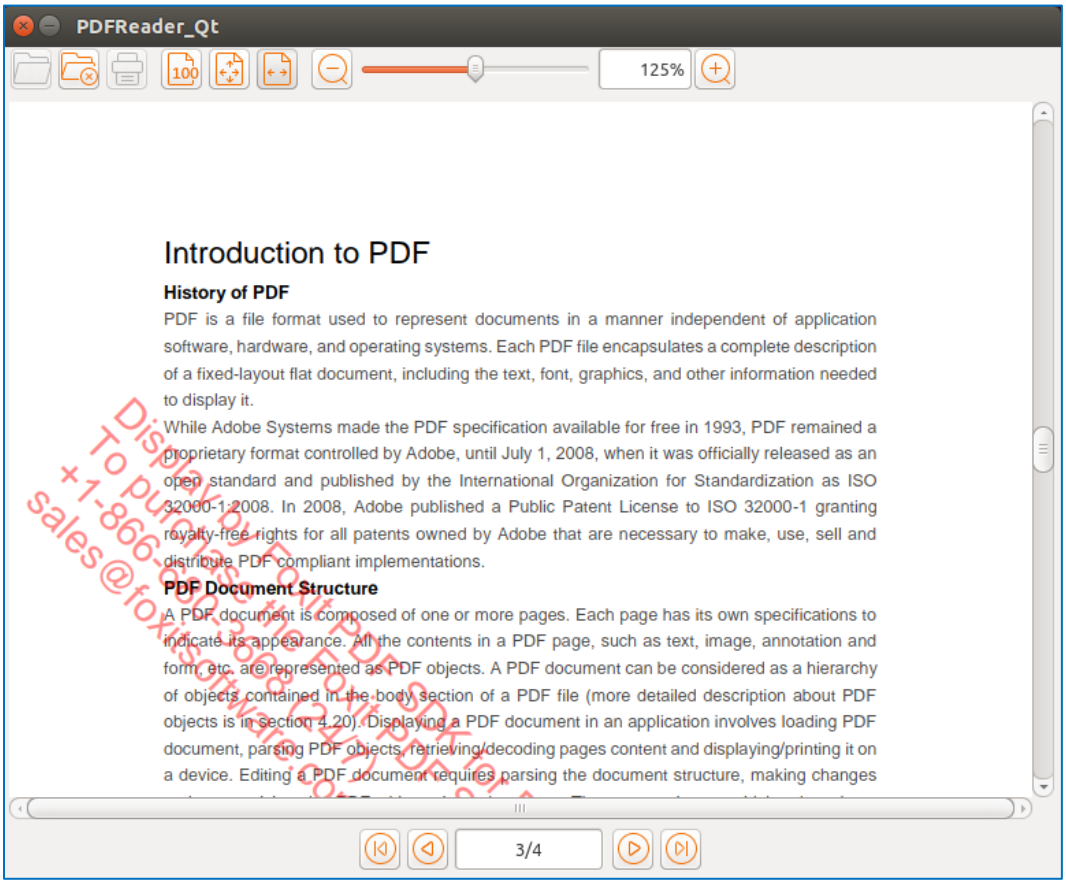


Figure 2-8

2.3.1.3 创建一个简单的工程

本节主要介绍如何使用 Foxit PDF SDK for Linux 创建一个简单的工程，该工程将 PDF 文档的首页渲染成 bitmap，然后将其另存为 JPG 图片。请按照如下的步骤操作：

1) 新建一个名为 "test_linux" 的文件夹。

2) 将 "foxitpdfsdk_10_0_linux" 文件夹下的 "include" 和 "lib" 文件夹拷贝到 "test_linux" 文件夹下。

3) 在 "test_linux" 文件夹下，创建一个 "test_linux.cpp" 文件，然后添加代码，具体代码见 2.2.3 小节 "创建一个简单的工程" 中的 "test_win.cpp"。

   "test_linux.cpp" 如下所示：(为了更好的阅读，我们将代码拷贝到 Visual Studio 中以便使用不同的颜色显示代码)

```
#include <iostream>
```

```
#include "common/fs_common.h"
#include "pdf/fs_pdfdoc.h"
#include "pdf/fs_pdfpage.h"
#include "common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

int main(int argc, char* argv[])

{
	// 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=").
	const char* sn = " ";
	const char* key = " ";
	foxit::ErrorCode code = Library::Initialize(sn, key);
	if (code != foxit::e_ErrSuccess) {
		return FALSE;
	}

	// Load a PDF document, and parse the first page of the document.
	PDFDoc doc("../../Sample.pdf");
	ErrorCode error_code = doc.Load();
	if (error_code!= foxit::e_ErrSuccess) return 0;
	PDFPage page = doc.GetPage(0);
	page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

	int width = static_cast<int>(page.GetWidth());
	int height = static_cast<int>(page.GetHeight());
	Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

	// Prepare a bitmap for rendering.
	Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
	bitmap.FillRect(0xFFFFFFFF, NULL);
	// Render page.
	Renderer render(bitmap, false);
	render.StartRender(page, matrix, NULL);

	// Add the bitmap to image and save the image.
	Image img;
	img.AddFrame(bitmap);
	img.SaveAs("testpage.jpg");
	return 0;
}
```

4) 在 "test_linux" 文件夹下放入了一个 "Sample.pdf" 文档。

5) 创建一个 Makefile。在该 Makefile 中，构建路径应包含 PDF SDK 库。64 位系统使用 libfsdk_linux64.so，32 位系统使用 libfsdk_linux32.so。Makefile 文件示例如下所示：

```
CXX=g++

# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
FSDKLIB_PATH=-Llib
LIB_PATH=lib
FSDKLIB=-lfsdk_linux64
LIBNAME=libfsdk_linux64.so
LIBS=$(FSDKLIB) -lpthread
LDFLAGS=-Wl,--rpath=.
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c

# Copy the given library to the destination path
CP_LIB=cp $(LIB_PATH)/$(LIBNAME) $(DEST_PATH)

DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_linux

dir:
	mkdir -p $(DEST_PATH)
	mkdir -p $(OBJ_PATH)

test_linux.o :test_linux.cpp
	$(CXX) $(CCFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)

test_linux: dir test_linux.o
	$(CXX) $(OBJ_PATH)/test_linux.o $(DEST) $(FSDKLIB_PATH) $(LIBS) $(LDFLAGS)
```

6) 构建工程。打开终端，导航到 "test_linux"，然后运行 "**make test_linux**"以在 "test_linux/bin/rel_gcc" 文件夹下生成二进制文件。

7) 执行二进制文件。在终端中导航到二进制文件所在的目录，运行 "**./test_linux**"，然后在当前文件夹下会生成 "testpage.jpg"。

2.3.2 Linux (armv7/armv8)

从 8.0 版本开始，Foxit PDF SDK 提供了 Linux ARM 框架的 armv7 和 armv8 的 ".so" 库。

在本手册中，请知晓：图片上面高亮的矩形区域指的是 SDK 的版本号，当前 SDK 的版本是 10.0，则其代表 10_0。

2.3.2.1 包结构说明

下载 Foxit PDF SDK for Linux (arm)包，解压到一个新的目录如 "foxitpdfsdk_10_0_linux_arm"，如 Figure 2-9 所示。其中解压包中包括如下的内容：

doc: API 手册，开发者指南

examples: 示例工程和 demo

include: Foxit PDF SDK API 的头文件

lib: SDK 库和授权文件

res: 输出预览 (output preview) demo 使用的默认 icc profile 文件

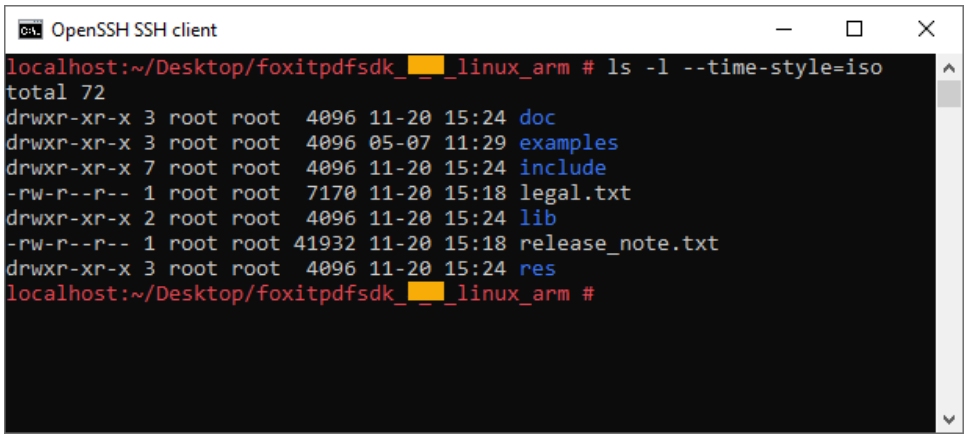


Figure 2-9

"examples/simple_demo" 包含了大量的 PDF 应用程序的 demo。

2.3.2.2 运行 demo

在运行 demo 之前，请确保您已经正确配置环境，并且已安装 CMake 3.1 或更高版本。

OS	Tool chain	GLIBC
Linux armv7	gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf 或更高版本	GLIBC_2.28 或更高版本
Linux armv8	gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu 或更高版本	GLIBC_2.27 或更高版本

Foxit PDF SDK 在 "examples/simple_demo" 目录下提供了一些简单示例 demo。除了 security, signature, office2pdf (armv8), pdf2office 和 ofd (armv8) 以外，其他 demo 都可以直接运行 "examples/simple_demo" 下的 ".sh" 文件：

- 在 "\examples\simple_demo" 目录下，运行 "**./RunAllDemo.sh all**"。
- 如果您只需要运行某个特定的 demo，使用 "**./RunAllDemo.sh demo_name**" 命令。比如，使用 "**./RunAllDemo.sh annotation**" 来运行 annotation demo。

对于有输出文件 (pdf, 文本或者图片文件) 的 demo 来说，会在 "examples/simple_demo/output_files/" 文件夹下生成以该 demo 名称命名的文件夹，并且输出文件将会在该文件夹下生成。

Security 和 Signature demo

在运行 **security** 和 **signature** demo 之前，请确保您已经安装了 OpenSSL。从 OpenSSL 官网下载 OpenSSL 源码包，或者您可以直接与我们客服联系。获取到源码包后，解压并进行如下操作：

1) 将 OpenSSL 文件夹拷贝到 "include"文件夹下，以确保 demo 中引用的 OpenSSL 头文件可以被识别到。
2) 将 "libssl.a" 和 "libcrypto.a" 库拷贝到 "lib" 文件夹下。
3) 运行 demo。

备注：*OpenSSL 1.0.2 版本在 security 和 signature demo 中已经验证是可用的。您可以替换为其他所需的版本，但可能需要做一些相应的更改。*

Office to PDF demo(armv8)

对于 **office2pdf demo**，请参考 3.39 小节 "Office 转 PDF"。

PDF to Office demo

对于如何运行 **pdf2office** demo，请参考 3.45 小节 "PDF 转 Office"。

OFD demo (armv8)

对于如何运行 **ofd** demo，请参考 3.47 小节 "OFD"。

2.3.2.3 创建一个简单的工程

使用 Foxit PDF SDK for Linux (arm) 创建一个简单的工程与 Linux (x86/x64) 中的创建一个简单的工程 是类似的，只是 Makefile 不一样。

在 Makefile 中，构建路径应包含 PDF SDK 库。根据您的系统架构，使用 **libfsdk_linuxarmv7.so** 或者 **libfsdk_linuxarmv8.so**。Makefile 文件示例如下所示：

```
CXX=g++

# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
FSDKLIB_PATH=-Llib
LIB_PATH=lib
```

```
FSDKLIB=-libfsdk_linuxarmv8.so
LIBNAME=libfsdk_linuxarmv8.so
LIBS=$(FSDKLIB) -lpthread
LDFLAGS=-Wl,--rpath=.
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c

# Copy the given library to the destination path
CP_LIB=cp $(LIB_PATH)/$(LIBNAME) $(DEST_PATH)

DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_linux

dir:
	mkdir -p $(DEST_PATH)
	mkdir -p $(OBJ_PATH)

test_linux.o :test_linux.cpp
	$(CXX) $(CCFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)

test_linux: dir test_linux.o
	$(CXX) $(OBJ_PATH)/test_linux.o $(DEST) $(FSDKLIB_PATH) $(LIBS) $(LDFLAGS)
```

构建工程。打开终端，导航到 "test_linux"，然后运行 "**make test_linux**"以在 "test_linux/bin/rel_gcc" 文件夹下生成二进制文件。

执行二进制文件。在终端中导航到二进制文件所在的目录，运行 "**./test_linux**"，然后在当前文件夹下会生成 "testpage.jpg"。

2.4 Mac

2.4.1 Mac for x64

在本手册中，请知晓：图片上面高亮的矩形区域指的是 SDK 的版本号，当前 SDK 的版本是 10.0，则其代表 10_0。

2.4.1.1 包结构说明

下载 Foxit PDF SDK for Mac (x64) 包，解压到一个新的目录如 "foxitpdfsdk_10_0_mac"，如 Figure 2-10 所示。其中解压包中包括如下的内容：

doc: API 手册，开发者指南

examples: 示例工程和 demo

include: Foxit PDF SDK API 的头文件

lib: SDK 库和授权文件

res: 输出预览 (output preview) demo 使用的默认 icc profile 文件

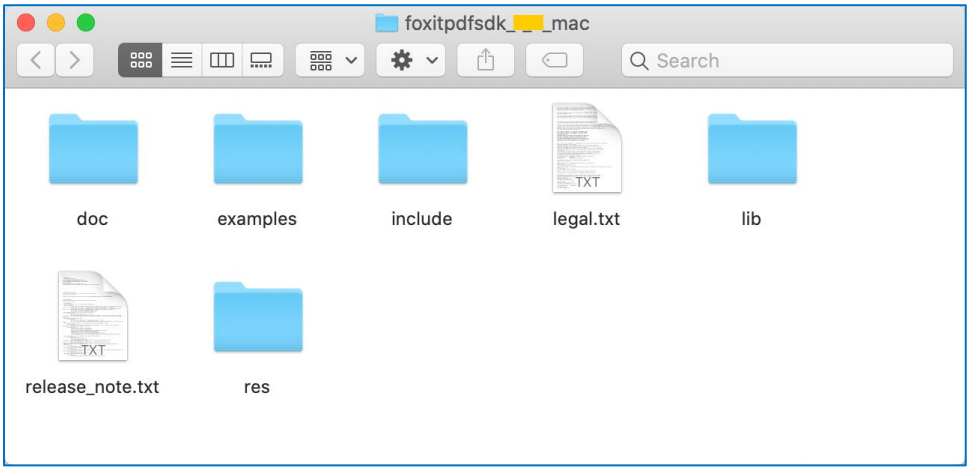


Figure 2-10

在 "\examples\simple_demo" 文件夹下，包含了大量的 PDF 应用程序的 demo。

2.4.1.2 运行 demo

备注：从 9.0 版本开始，用于构建和编译 Foxit PDF SDK for Mac (x64) 的 clang 版本从 9.1.0 升级到 11.0.3。

在运行 demo 之前，请确保您已经正确配置环境，并且已安装 CMake 3.1 或更高版本。

OS	编译工具或 IDE	版本
Mac	Xcode	8 或更高

在终端运行 demo (除了 security, signature, compliance, html2pdf, output preview 和 dwg2pdf demo，这些将在后面介绍)，您可以按照如下的步骤：

1) 打开一个终端，导航到 "foxitpdfsdk_10_0_mac/examples/simple_demo";
2) 运行 "**cmake -DPRJ_NAME=XXX**" 以编译指定的 demo。"XXX" 是 demo 的名称，该名称必须为 "annotation", "attachment", "pdf2text" 等等。例如，运行 "cmake -DPRJ_NAME=annotation"。
3) 运行 "**make**" 以构建上述命令中指定的 demo。然后将生成名为 "**XXX_xxx**" 的可执行文件。"XXX" 是 demo 的名称，"xxx" 是架构名，例如 "annotation_mac64"。
4) 执行 "**./XXX_xxx**" 以运行 demo。例如，执行 "./annotation_mac64" 来运行 annotation demo。

"examples/simple_demo/input_files" 包含了所有 demo 中需要使用的测试文档。对于有输出文件 (pdf, 文本或者图片文件) 的 demo 来说，会在 "examples/simple_demo/output_files/" 文件夹下生成以该 demo 名称命名的文件夹，并且输出文件将会在该文件夹下生成。

Security 和 Signature demo

在运行 **security** 和 **signature** demo 之前，请确保您已经安装了 OpenSSL。从 OpenSSL 官网下载 OpenSSL 源码包，或者您可以直接与我们客服联系。获取到源码包后，解压并进行如下操作：

1) 将 OpenSSL 文件夹拷贝到 "include"文件夹下，以确保 demo 中引用的 OpenSSL 头文件可以被识别到。
2) 将 "libssl.a" 和 "libcrypto.a" 库拷贝到 "lib" 文件夹下。
3) 参考其他 demo 的运行步骤运行该 demo。

备注：*OpenSSL 1.0.2 版本在 security 和 signature demo 中已经验证是可用的。您可以替换为其他所需的版本，但可能需要做一些相应的更改。*

Compliance demo

对于如何运行 **compliance** demo，请参阅 3.35 小节 "Compliance"。

HTML to PDF demo

对于如何运行 **html2pdf** demo，请参阅 3.37 小节 "HTML 转 PDF"。

Output Preview demo

对于如何运行 **output preview** demo，请参考 3.39 小节 "输出预览 (Output Preview)"。

Dwg to PDF demo

对于如何运行 **dwg2pdf** demo，请参考 3.46 小节 "DWG 转 PDF"。

2.4.1.3 创建一个简单的工程

本节主要介绍如何使用 Foxit PDF SDK for Mac x64 (C++) 创建一个简单的工程，该工程将 PDF 文档的首页渲染成 bitmap，然后将其另存为 JPG 图片。请按照如下的步骤操作：

1) 新建一个名为 "test_mac" 的文件夹。
2) 将 "foxitpdfsdk_10_0_mac" 文件夹下的 "include" 和 "lib" 文件夹拷贝到 "test_mac" 文件夹下。
3) 在 "test_mac" 文件夹下，创建一个 "test_mac.cpp" 文件，然后添加代码，具体代码见 2.2.3 小节 "创建一个简单的工程" 中的 "test_win.cpp"。

   "test_mac.cpp" 如下所示：

```
#include <iostream>
#include "common/fs_common.h"
#include "pdf/fs_pdfdoc.h"
#include "pdf/fs_pdfpage.h"
#include "common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

int main(int argc, char* argv[])

{
   // 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=").
   const char* sn = " ";
   const char* key = " ";
   foxit::ErrorCode code = Library::Initialize(sn, key);
   if (code != foxit::e_ErrSuccess) {
      return FALSE;
   }

   // Load a PDF document, and parse the first page of the document.
   PDFDoc doc("../../Sample.pdf");
   ErrorCode error_code = doc.Load();
   if (error_code!= foxit::e_ErrSuccess) return 0;
   PDFPage page = doc.GetPage(0);
   page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

   int width = static_cast<int>(page.GetWidth());
   int height = static_cast<int>(page.GetHeight());
   Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

   // Prepare a bitmap for rendering.
   Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
   bitmap.FillRect(0xFFFFFFFF, NULL);
   // Render page.
   Renderer render(bitmap, false);
   render.StartRender(page, matrix, NULL);

   // Add the bitmap to image and save the image.
   Image img;
   img.AddFrame(bitmap);
   img.SaveAs("testpage.jpg");
   return 0;
}
```

4) 在 "test_mac" 文件夹下放入了一个 "Sample.pdf" 文档。

5) 创建一个 Makefile。在该 Makefile 中，构建路径应包含 PDF SDK 库。使用 libfsdk_mac64.dylib。Makefile 文件示例如下所示：

```
CXX=g++

# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
LIBNAME=./lib/libfsdk_mac64.dylib
LDFLAGS=-Wl,-rpath,../../lib
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c

DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_mac

dir:
	mkdir -p $(DEST_PATH)
	mkdir -p $(OBJ_PATH)

test_mac.o :test_mac.cpp
	$(CXX) $(CCFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)

test_mac: dir test_mac.o
	$(CXX) $(OBJ_PATH)/test_mac.o $(DEST) $(LDFLAGS) $(LIBNAME)
```

6) 构建工程。打开终端，导航到 "test_mac"，然后运行 "**make test_mac**"以在 "test_mac/bin/rel_gcc" 文件夹下生成二进制文件。

7) 执行二进制文件。在终端中导航到二进制文件所在的目录，运行 "**./test_mac**"，然后在当前文件夹下会生成 "testpage.jpg"。

2.4.2 Mac for arm64

从 8.4 版本开始，Foxit PDF SDK 支持 Mac ARM64 框架，提供了 arm64 的 SDK 库。

在本手册中，请知晓：图片上面高亮的矩形区域指的是 SDK 的版本号，当前 SDK 的版本是 10.0，则其代表 10_0。

2.4.2.1 包结构说明

下载 Foxit PDF SDK for Mac (arm64) 包，解压到一个新的目录如 "foxitpdfsdk_10_0_mac_arm64"，如 Figure 2-11 所示。其中解压包中包括如下的内容：

doc: API 手册，开发者指南

examples: 示例工程和 demo

include: Foxit PDF SDK API 的头文件

lib: SDK 库和授权文件

res: 输出预览 (output preview) demo 使用的默认 icc profile 文件

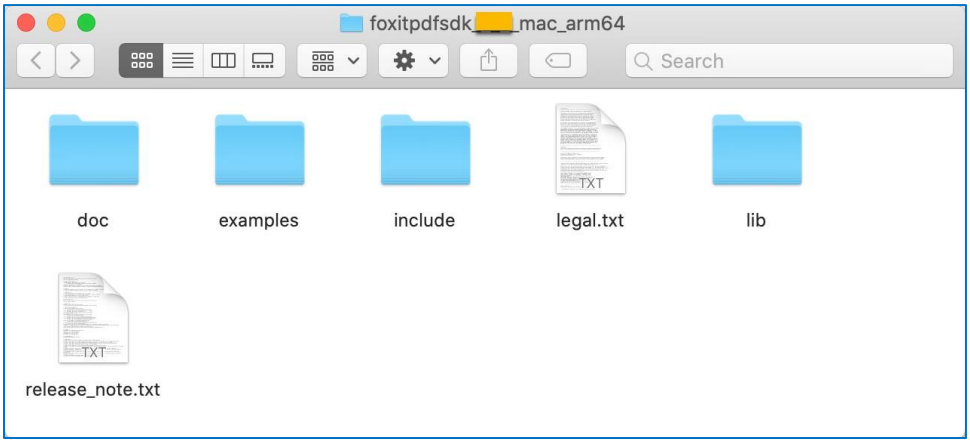


Figure 2-11

在 "\examples\simple_demo" 文件夹下，包含了大量的 PDF 应用程序的 demo。

2.4.2.2 运行 demo

在运行 demo 之前，请确保您已经正确配置环境，并且已安装 CMake 3.2 或更高版本。

OS	编译工具或 IDE	版本
Mac	Xcode	13 或更高

在终端运行 demo (除了 security, signature, compliance 和 html2pdf，这些将在后面介绍)，您可以按照如下的步骤：

1) 打开一个终端，导航到 "foxitpdfsdk_10_0_mac_arm64/examples/simple_demo";
2) 运行 "**cmake -DPRJ_NAME=XXX**" 以编译指定的 demo。"XXX" 是 demo 的名称，该名称必须为 "annotation", "attachment", "pdf2text" 等等。例如，运行 "cmake -DPRJ_NAME=annotation"。
3) 运行 "**make**" 以构建上述命令中指定的 demo。然后将生成名为 "**XXX_xxx**" 的可执行文件。"XXX" 是 demo 的名称，"xxx" 是架构名，例如 "annotation_mac64"。
4) 执行 "**./XXX_xxx**" 以运行 demo。例如，执行 "./annotation_mac64" 来运行 annotation demo。

"examples/simple_demo/input_files" 包含了所有 demo 中需要使用的测试文档。对于有输出文件 (pdf, 文本或者图片文件) 的 demo 来说，会在 "examples/simple_demo/output_files/" 文件夹下生成以该 demo 名称命名的文件夹，并且输出文件将会在该文件夹下生成。

Security 和 Signature demo

在运行 **security** 和 **signature** demo 之前，请确保您已经安装了 OpenSSL。从 OpenSSL 官网下载 OpenSSL 源码包，或者您可以直接与我们客服联系。获取到源码包后，解压并进行如下操作：

1) 将 OpenSSL 文件夹拷贝到 "include"文件夹下，以确保 demo 中引用的 OpenSSL 头文件可以被识别到。
2) 将 "libssl.a" 和 "libcrypto.a" 库拷贝到 "lib" 文件夹下。
3) 参考其他 demo 的运行步骤运行该 demo。

备注：*OpenSSL 1.1.1-stable 版本在 security 和 signature demo 中已经验证是可用的。您可以替换为其他所需的版本，但可能需要做一些相应的更改。*

Compliance demo

对于如何运行 **compliance** demo，请参阅 3.35 小节 "Compliance"。

HTML to PDF demo

关于如何运行该 demo 的更详细的信息，请参考 3.37 小节 "HTML 转 PDF"。

2.4.2.3 创建一个简单的工程

本节主要介绍如何使用 Foxit PDF SDK for Mac arm64 (C++) 创建一个简单的工程，该工程将 PDF 文档的首页渲染成 bitmap，然后将其另存为 JPG 图片。请按照如下的步骤操作：

1) 新建一个名为 "test_mac_arm64" 的文件夹。
2) 将 "foxitpdfsdk_10_0_mac_arm64" 文件夹下的 "include" 和 "lib" 文件夹拷贝到 "test_mac_arm64" 文件夹下。
3) 在 "test_mac_arm64" 文件夹下，创建一个 "test_mac_arm64.cpp" 文件，然后添加代码，具体代码见 2.2.3 小节 "创建一个简单的工程" 中的 "test_win.cpp"。

   "test_mac_arm64.cpp" 如下所示：

```
#include <iostream>
#include "common/fs_common.h"
#include "pdf/fs_pdfdoc.h"
#include "pdf/fs_pdfpage.h"
#include "common/fs_render.h"

using namespace std;
```

```
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

int main(int argc, char* argv[])

{
   // 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=").
   const char* sn = " ";
   const char* key = " ";
   foxit::ErrorCode code = Library::Initialize(sn, key);
   if (code != foxit::e_ErrSuccess) {
      return FALSE;
   }

   // Load a PDF document, and parse the first page of the document.
   PDFDoc doc("../../Sample.pdf");
   ErrorCode error_code = doc.Load();
   if (error_code!= foxit::e_ErrSuccess) return 0;
   PDFPage page = doc.GetPage(0);
   page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

   int width = static_cast<int>(page.GetWidth());
   int height = static_cast<int>(page.GetHeight());
   Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

   // Prepare a bitmap for rendering.
   Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
   bitmap.FillRect(0xFFFFFFFF, NULL);
   // Render page.
   Renderer render(bitmap, false);
   render.StartRender(page, matrix, NULL);

   // Add the bitmap to image and save the image.
   Image img;
   img.AddFrame(bitmap);
   img.SaveAs("testpage.jpg");
   return 0;
}
```

4) 在 "test_mac_arm64" 文件夹下放入了一个 "Sample.pdf" 文档。

5) 创建一个 Makefile。在该 Makefile 中，构建路径应包含 PDF SDK 库。使用 libfsdk_macarm.dylib。Makefile 文件示例如下所示：

```
CXX=g++
```

```
# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
LIBNAME=./lib/libfsdk_macarm.dylib
LDFLAGS=-Wl,-rpath,../../lib
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c
CXXFLAGS=-std=c++11
DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_mac_arm64
dir:
	mkdir -p $(DEST_PATH)
	mkdir -p $(OBJ_PATH)

test_mac_arm64.o :test_mac_arm64.cpp
	$(CXX) $(CCFLAGS) $(CXXFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)

test_mac_arm64: test_mac_arm64.o
	$(CXX) $(OBJ_PATH)/test_mac_arm64.o $(DEST) $(LDFLAGS) $(LIBNAME)
```

6) 构建工程。打开终端，导航到 "test_mac_arm64"，然后运行 "**make test_mac_arm64**"以在 "test_mac_arm64/bin/rel_gcc" 文件夹下生成二进制文件。

7) 执行二进制文件。在终端中导航到二进制文件所在的目录，运行 "**./test_mac_arm64**"，然后在当前文件夹下会生成 "testpage.jpg"。

3 使用 SDK API

在本节中，我们将介绍 Foxit PDF SDK 的主要功能，并列举相关示例来展示如何使用 Foxit PDF SDK C++ API 将强大的 PDF 功能集成到您的应用程序中。您可以参阅 API reference [2] 来获取示例中 APIs 更详细的使用说明。

3.1 初始化库

在调用任何 API 之前，都需要首先初始化 Foxit PDF SDK。foxit::common::Library::Initialize 用来初始化 Foxit PDF SDK，您需要购买正式的 license 来获取 license key 和序列号。当不再需要使用 Foxit PDF SDK 时，请调用 foxit::common::Library::Release 将其释放。

***备注**：参数 "sn" 的值在 "**gsdk_sn.txt**" 中 ("SN=" 后面的字符串)，"key" 的值在 "**gsdk_key.txt**" 中 ("Sign=" 后面的字符串)。*

Example:

3.1.1 如何初始化 Foxit PDF SDK

```
#include "include/common/fs_common.h"

using namespace foxit;
using namespace common;
...

const char* sn = " ";
const char* key = " ";
foxit::ErrorCode code = Library::Initialize(sn, key);
if (code != foxit::e_ErrSuccess) {
        return FALSE;
}
```

3.2 文档 (Document)

一个 PDF document 对象可以由一个已有的 PDF 文件从文件路径、内存缓冲区、自定义实现的 ReaderCallback 对象、输入文件流中构建。然后调用 PDFDoc::Load 或者 PDFDoc::StartLoad 加载文档内容。PDF document 对象用于文档级操作，比如打开和关闭 PDF 文档，获取页面、metadata 等。

Example:

3.2.1 如何从 0 开始创建一个 PDF 文档

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
```

```
using namespace common;
using namespace pdf;
...

PDFDoc doc();
```

备注：*创建一个新的PDF文档，该文档没有任何页面。*

3.2.2 如何通过文件路径加载一个现有的 PDF 文档

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.3 如何通过内存缓冲区加载一个现有的 PDF 文档

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

FILE* pFile = fopen(TEST_DOC_PATH"blank.pdf", "rb");
ASSERT_EQ(TRUE, NULL != pFile);
fseek(pFile, 0, SEEK_END);
long lFileSize = ftell(pFile);
char* buffer = new char[lFileSize];
memset(buffer, 0, sizeof(char)*lFileSize);
fseek(pFile, 0, SEEK_SET);
fread(buffer, sizeof(char), lFileSize, pFile);
fclose(pFile);
PDFDoc doc = PDFDoc(buffer, lFileSize);
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.4 如何通过自定义实现的 ReaderCallback 对象加载一个现有的 PDF 文档

```
#include "include/pdf/fs_pdfdoc.h"

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

```
class CFSFile_Read : public ReaderCallback
{
public:
	CFSFile_Read():m_fileFP(NULL)
		,m_bLargeFile(FALSE)
	{}

	~CFSFile_Read() {}

	bool LoadFile(const wchar_t* wFilePath, bool bLargeFile = FALSE)
	{
		std::wstring strTemp(wFilePath);
		string bstrFilepath = wchar2utf8(strTemp.c_str(), strTemp.size());

		m_fileFP = fopen(bstrFilepath.c_str(), "rb");
		if (!m_fileFP) return FALSE;

		m_bLargeFile = bLargeFile;
		return TRUE;
	}

	bool LoadFile(const char* filePath, bool bLargeFile = FALSE)
	{
		m_fileFP = fopen(filePath, "rb");
		if (!m_fileFP) return FALSE;

		m_bLargeFile = bLargeFile;
		return TRUE;
	}

	FILESIZE GetSize()
	{
		if (m_bLargeFile)
		{
#if defined(_WIN32) || defined(_WIN64)
			_fseeki64(m_fileFP, 0, SEEK_END);
			long long sizeL = _ftelli64(m_fileFP);
#elif defined(__linux__) || defined(__APPLE__)
			fseeko(m_fileFP, 0, SEEK_END);
			long long sizeL = ftello(m_fileFP);
#endif
			return sizeL;
		}
		else
		{
			fseek(m_fileFP, 0, SEEK_END);
			return (uint32)ftell(m_fileFP);
		}
	}

	int ReadBlock(void* buffer, FILESIZE offset, size_t size)
```

```
	{
		if (m_bLargeFile)
		{
#if defined(_WIN32) || defined(_WIN64)
			_fseeki64(m_fileFP, offset, SEEK_SET);
#elif defined(__linux__) || defined(__APPLE__)
			fseeko(m_fileFP, offset, SEEK_SET);
#endif
			long long readSize = fread(buffer, 1, size, m_fileFP);
			return (readSize == size);
		}
		else
		{
			if (!m_fileFP) return false;
			if(0 != fseek(m_fileFP, offset, 0))
				return false;
			if(0 == fread(buffer, size, 1, m_fileFP))
				return false;
			return true;
		}
	}

	size_t	ReadBlock(void* buffer, size_t size) {
		if (m_bLargeFile)
		{
#if defined(_WIN32) || defined(_WIN64)
			_fseeki64(m_fileFP, 0, SEEK_SET);
#elif defined(__linux__) || defined(__APPLE__)
			fseeko(m_fileFP, 0, SEEK_SET);
#endif
			return fread(buffer, 1, size, m_fileFP);
		}
		else
		{
			if (!m_fileFP) return false;
			if(0 != fseek(m_fileFP, 0, 0))
				return 0;
			return fread(buffer, size, 1, m_fileFP);
		}
	}

	void Release()
	{
		if(m_fileFP)
			fclose(m_fileFP);
		m_fileFP = NULL;
		delete this;
	}

private:
	FILE* m_fileFP;
	bool m_bLargeFile;
```

```
}
...
string inputPDFPath = "Sample.pdf";
CFSFile_Read* pFileRead = new CFSFile_Read();
If(!pFileRead->LoadFile(inputPDFPath.c_str()))
        Return;
PDFDoc doc = PDFDoc(pFileRead);
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.5 如何加载 PDF 文档以及获取文档的首页

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
```

3.2.6 如何将 PDF 文档另存为一个新的文档

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
doc.SaveAs("new_Sample.pdf", PDFDoc::e_SaveFlagNoOriginal);
```

3.2.7 如何通过 WriterCallback 将 PDF 文档保存到内存缓冲区

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

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

```
// FileWriter for saving file to memory buffer.
class  FileWriter:public common::file::WriterCallback
{
public:
        FileWriter() {}

        ~FileWriter()  {}

        FILESIZE GetSize() {
                return  binary_buffer_.GetSize();
        }

        FX_BOOL Flush() {
                return  TRUE;
        }

        FX_BOOL WriteBlock(const void* buffer,FILESIZE offset,size_t size) {
                return  binary_buffer_.InsertBlock(offset,buffer,size);
        }

        FX_BOOL  ReadBlock(void* buffer,FILESIZE offset,size_t size)    {
                FX_LPBYTE byte_buffer = binary_buffer_.GetBuffer();
                memcpy(buffer, byte_buffer + offset, size);
        }

        void Release() {
        }

        CFX_BinaryBuf GetBuffer() {
                return binary_buffer_;
        }
private:
        CFX_BinaryBuf binary_buffer_;
};
...

FileWriter* filewriter = new FileWriter();

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

doc.StartSaveAs(filewriter, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.3 页面 (Page)

PDF 页面是 PDF Document 基础和重要的组成部分。使用函数 foxit::pdf::PDFDoc::GetPage 从文档中获取 foxit::pdf::PDFPage 对象。页面级 API 提供了解析/渲染/编辑 (包括创建、删除、扁平化等) 页面、获取 PDF 注释、获取和设置页面属性等功能。对于大多数情况，在渲染和处理页面之前，需要先对页面进行解析。

Example:

3.3.1 如何获取页面的大小

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
```

3.3.2 如何计算页面内容的边界框

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
//Assuming PDFDoc doc has been loaded.
//Assuming PDFPage page has been loaded and parsed.

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

3.3.3 如何创建一个 PDF 页面以及设置其页面大小

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

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

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

3.3.4 如何删除一个 PDF 页面

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace 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 如何扁平化一个 PDF 页面

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

// Flatten all contents of a PDF page.
page.Flatten(true, PDFPage::e_FlattenAll);
```

```
// Flatten a PDF page without annotations.
page.Flatten(true, PDFPage::e_FlattenNoAnnot);

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

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

3.3.6 如何获取和设置 PDF 文档中的页面缩略图

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

Bitmap bmp();
// Write bitmap data to the bmp object.
...
// Set thumbnails to the page.
page.SetThumbnail(bmp);
// Load thumbnails in the page.
```

```
Bitmap bitmap = page.LoadThumbnail();
...
```

3.4 渲染 (Render)

PDF 渲染是通过 Foxit 渲染引擎实现的，Foxit 渲染引擎是一个图形引擎，用于将页面渲染到位图或平台设备上下文。 Foxit PDF SDK 提供了 APIs 用来设置渲染选项/flags，例如设置 flag 来决定是否渲染表单域和签名，是否绘制图像反锯齿 (anti-aliasing) 和路径反锯齿。可以使用以下 APIs 进行渲染：

- 渲染页面和注释时，首先使用 Renderer::SetRenderContentFlags 接口来决定是否同时渲染页面和注释，然后使用 Renderer::StartRender 接口进行渲染。Renderer::StartQuickRender 接口也可以用来渲染页面，但仅用于缩略图。
- 渲染单个 annotation 注释，使用 Renderer::RenderAnnot 接口。
- 在位图上渲染，使用 Renderer::StartRenderBitmap 接口。
- 渲染一个重排的页面，使用 Renderer::StartRenderReflowPage 接口。

在 Foxit PDF SDK 中，Widget 注释常与表单域和表单控件相关联。渲染 widget 注释，推荐使用如下的流程：

- 加载 PDF 页面后，首先渲染页面以及该页面上所有的注释 (包括 widget 注释)。
- 然后，如果使用 pdf::interform::Filler 对象来填表，则应使用 pdf::interform::Filler::Render 接口来渲染当前获取到焦点的表单控件，而不是使用 Renderer::RenderAnnot 接口。

Example:

3.4.1 如何将 PDF 页面渲染到 bitmap

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFPage page has been loaded and parsed.

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);
// Render page.
```

```
Renderer render(bitmap, false);
render.StartRender(page, matrix, NULL);
...
```

3.4.2 如何渲染页面和注释

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFPage page has been loaded and parsed.
int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);

Renderer render(bitmap, false);
uint32 dwRenderFlag = Renderer::e_RenderAnnot | Renderer::e_RenderPage;
render.SetRenderContentFlags(dwRenderFlag);
render.StartRender(page, matrix, NULL);
...
```

3.5 附件 (Attachment)

在 Foxit PDF SDK 中，attachments 指的是文档附件而不是文件附件注释。它允许将整个文件封装在文档中，就像电子邮件附件一样。Foxit PDF SDK 提供应用程序 APIs 来访问附件，例如加载附件，获取附件，插入/删除附件，以及访问附件的属性。

Example:

3.5.1 如何从 PDF 文档中导出嵌入的附件文件，并将其另存为单个文件

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfattachments.h"

using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;
```

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

// Get information of attachments.
Attachments attachments(doc);
int count = attachments.GetCount();
for (int i = 0; i < count; i++) {
   WString key = attachments.GetKey(i);
   FileSpec file_spec = attachments.GetEmbeddedFile(key);
   if (!file_spec.IsEmpty()) {
          WString name = file_spec.GetFileName();
          if (file_spec.IsEmbedded()) {
             WString exFilePath = "output_directory";
             file_spec.ExportToFile(exFilePath);
          }
   }
}
...
```

3.5.2 如何删除 PDF 文档中所有的附件文件

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfattachments.h"
...
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFDoc doc has been loaded.

// Get information of attachments.
Attachments attachments(doc);
int count = attachments.GetCount();
for (int i = 0; i < count; i++) {
    WString key = attachments.GetKey(i);
    attachment.RemoveEmbeddedFile(&key);
}
...
```

3.6 文本页面 (Text Page)

Foxit PDF SDK 提供 APIs 来提取，选择，搜索和检索 PDF 文档中的文本。 PDF 文本内容存储在与特定页面相关的 TextPage 对象中。TextPage 类可用于获取 PDF 页面中文本的信息，例如单个字符，单个单词，指定字符范围或矩形内的文本内容等。它还可用于构造其他文本相关类的对象，用来对文本内容执行更多操作或从文本内容访问指定信息：

- 在 PDF 页面的文本内容中搜索文本，使用 TextPage 对象来构建 TextSearch 对象。

- 访问类似超文本链接的文本，使用 TextPage 对象来构建 PageTextLinks 对象。

Example:

3.6.1 如何从 PDF 页面中提取文本

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_search.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
TextPage text_page(page);
int count = text_page.GetCharCount();
if (count > 0) {
    WString text = text_page.GetChars();
    String s_text = text.UTF8Encode();
    fwrite((const char*)s_text, sizeof(char), s_text.GetLength(), file);
}
...
```

3.6.2 如何在 PDF 文档中获取矩形区域中的文本

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

RectF rect;
rect.left = 90;
rect.right = 450;
rect.top = 595;
rect.bottom = 580;
TextPage textPage = new TextPage (&page, TextPage::e_ParseTextNormal);
textPage.GetTextInRect(&rect);
...
```

3.7 文本搜索 (Text Search)

Foxit PDF SDK 提供 APIs 来搜索 PDF 文档、XFA 文档、文本页面或者 PDF 注释中的文本。它提供了文本搜索和获取搜索结果的函数：

- 指定搜索模式和选项，使用 TextSearch::SetPattern、TextSearch::SetStartPage (仅对 PDF 文档中的文本搜索有用)、TextSearch::SetEndPage (仅对 PDF 文档中的文本搜索有用)、和 TextSearch::SetSearchFlags 接口。
- 进行搜索，使用 TextSearch::FindNext 和 TextSearch::FindPrev 接口。
- 获取搜索结果，使用 TextSearch::GetMatchXXX() 接口。

Example:

3.7.1 如何在 PDF 文档中搜索指定的文本

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

// Search for all pages of doc.
TextSearch search(doc, NULL);

int start_index = 0, end_index = doc.GetPageCount() - 1;
search.SetStartPage(start_index);
search.SetEndPage(end_index);

WString pattern = L"Foxit";
search.SetPattern(pattern);

foxit::uint32 flags = TextSearch::e_SearchNormal;
search.SetSearchFlags(flags);
...

int match_count = 0;
while (search.FindNext()) {
    RectFArray rect_array = search.GetMatchRects();
    match_count ++;
  }
...
```

3.8 搜索和替换 (Search and Replace)

搜索和替换功能允许您在 PDF 文档中查找特定的文本内容，并用新的内容替换它。

3.8.1 系统需求

平台: Windows, Linux, Mac

开发语言: C, C++, Java, C#, Python, Objective-C

License Key: license key 中包含 'AdvEdit' 模块的权限

SDK 版本: Foxit PDF SDK 9.0 或更高版本

3.8.2 如何使用搜索和替换功能

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/addon/pageeditor/fs_searchreplace.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace foxit::addon::pageeditor;

PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();

// Instantiate a TextSearchReplace object.
TextSearchReplace text_searchreplace(doc);

// Configure search options, match whole words only, whether to set match only whole words and match case.
FindOption find_option(true, true);

ReplaceCallbackImpl* replace_callback = new ReplaceCallbackImpl();
// Set replacing callback function.
text_searchreplace.SetReplaceCallback(replace_callback);

// Set keywords and page index to do searching and replacing.
text_searchreplace.SetPattern(L"PDF", 0, find_option);

// Replace with new text.
while (text_searchreplace.ReplaceNext(L"PDC")) {}
```

3.9 文本链接 (Text Link)

在 PDF 页面中，指向网站、网络资源以及电子邮件地址的超链接文本和普通文本一样。在处理文本链接之前，用户应首先调用 PageTextLinks::GetTextLink 接口来获取一个 textlink 对象。

Example:

3.9.1 如何检索 PDF 页面中的超链接

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
TextPage text_page(page);
PageTextLinks pageTextLink(text_page);
TextLink textLink = pageTextLink.GetTextLink(index);
String strURL = textLink.GetURI();
...
```

3.10 书签 (Bookmark)

Foxit PDF SDK 提供了名为书签的导航工具，允许用户在 PDF 文档中快速定位和链接他们感兴趣的部分。PDF 书签也称为大纲 (outline)，每个书签包含一个目标位置或动作来描述它链接到的位置。它是一个树形的层次结构，因此在访问 bookmark 树之前，必须首先调用接口 pdf::PDFDoc::GetRootBookmark 以获取整个 bookmark 树的根节点。这里，"书签根节点" 是一个抽象对象，它只有一些子节点，没有兄弟节点, 也没有任何数据 (包括 bookmark 数据，目标位置数据和动作数据)。因为它没有任何数据，因此无法在应用程序界面上显示，能够调用的接口只有 Bookmark::GetFirstChild。

在获取书签根节点后，就可以调用以下的接口去访问其他的书签:

- 访问 parent bookmark，使用 Bookmark::GetParent 接口。
- 访问第一个 child bookmark，使用 Bookmark::GetFirstChild 接口。
- 访问 next sibling bookmark，使用 Bookmark::GetNextSibling 接口。
- 插入一个新的 bookmark，使用 Bookmark::Insert 接口。
- 移动一个 bookmark，使用 Bookmark::MoveTo 接口。

Example:

3.10.1 如何遍历 PDF 文档中所有的书签

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_filespec.h"
```

```
#include "include/pdf/fs_bookmark.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// 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();
                  }
      }
}
...
```

3.10.2 如何向 PDF 文档中插入一个新的书签

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_filespec.h"
#include "include/pdf/fs_bookmark.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;

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

```
Bookmark root = doc.GetRootBookmark();
if (root.IsEmpty())
{
  root = doc.CreateRootBookmark();
}
Destination dest = Destination::CreateFitPage(doc, 0);
CFX_WideString ws_title;
ws_title.Format((FX_LPCWSTR)L"A bookmark to a page (index: %d)", 0);
Bookmark child = root.Insert(ws_title, foxit::pdf::Bookmark::e_PosLastChild);
child.SetDestination(dest);
child.SetColor(0xF68C21);
```

3.10.3 如何根据 PDF 的书签信息创建目录表

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_filespec.h"
#include "include/pdf/fs_bookmark.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

void AddTOCToPDF(PDFDoc doc) {
   //Set the table of contents configuration.
   Int32Array intarray;
   int depth = doc.GetBookmarkLevelDepth();
   if (depth > 0) {
      for (int i = 1; i <= depth; i++) {
         intarray.Add(i);
      }
   }
   WString title = L"";
   TableOfContentsConfig toc_config = TableOfContentsConfig(title, intarray, true, false);

   //Add the table of contents
   doc.AddTableOfContents(toc_config);
}
```

3.11 表单 (AcroForm)

PDF 目前支持两种类型的表单，用于以交互方式收集用户信息：AcroForms 和 XFA 表单。
Acroforms 是基于 PDF 框架的原始的可填写表单。Foxit PDF SDK 提供了以编程方式查看和编辑表单域的 APIs。在 PDF 文档中，表单域通常用于收集数据。Form 类提供了 APIs 用来获取表单域或表单控件，导入/导出表单数据，以及其他功能，例如：

- 获取表单域，使用 Form::GetFieldCount 和 Form::GetField 接口。

- 获取 PDF 页面中的表单控件，使用 Form::GetControlCount 和 Form::GetControl 接口。
- 从 XML 文件导入表单数据，使用 Form::ImportFromXML 接口；导出表单数据到 XML 文件，使用 Form::ExportToXML 接口。
- 获取 form filler 对象，使用 Form::GetFormFiller 接口。

从FDF/XFDF文件中导入表单数据，或者导出数据到FDF/XFDF文件，请参考 pdf::PDFDoc::ImportFromFDF 和 pdf::PDFDoc::ExportToFDF 接口。

Example:

3.11.1 如何加载 PDF 中的表单

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...

// Assuming PDFDoc doc has been loaded.

bool hasForm = doc.HasForm();
if(hasForm)
  Form form(doc);
...
```

3.11.2 如何获取表单域个数以及获取/设置其属性

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;

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

Form form(doc);
int countFields = form. GetFieldCount(NULL);
for (int i = 0; i < nFieldCount; i++)
{
    Field field = form.GetField(i, filter);
    Field::Type type = field.GetType();
    WString org_alternateName = field.GetAlternateName();
    field.SetAlternateName(L"signature");
}
```

3.11.3 如何将 PDF 中的表单数据导出到 XML 文件

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
...
form.ExportToXML(XMLFilePath);
...
```

3.11.4 如何通过 XML 文件导入表单数据到 PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
...
form.ImportFromXML(XMLFilePath);
...
```

3.11.5 如何获取表单域的坐标

1. 通过 PDFDoc 加载 PDF 文件。
2. 遍历 PDFDoc 的表单域，以获取表单的 field 对象。
3. 遍历 field 对象的 form controls，以获取 form control 对象。
4. 通过 form control 获取相关的 widget annotation 对象。
5. 调用 widget annotation 对象的 GetRect 方法获取表单的坐标。

```
#include <iostream>

#include "../../../include/common/fs_common.h"
#include "../../../include/pdf/fs_pdfdoc.h"
#include "../../../include/pdf/annots/fs_annot.h"
#include "../../../include/pdf/interform/fs_pdfform.h"

using namespace foxit;
```

```
using namespace foxit::common;
using namespace pdf;
using namespace annots;
using namespace interform;

...

PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
  printf("The Doc [%s] Error: %d\n", (const char*)String::FromUnicode(input_path), error_code);
  return 1;
}

if (!doc.HasForm()) return 1;
interform::Form form(doc);
for (int i = 0; i < form.GetFieldCount(NULL); i++) {
  interform::Field field = form.GetField(i, NULL);
  if (field.IsEmpty()) continue;
  for (int j = 0; j < field.GetControlCount(); j++) {
    interform::Control control = field.GetControl(j);
    annots::Widget widget = control.GetWidget();
    // Get rectangle of the  annot widget.
    RectF rect = widget.GetRect();
  }
}
```

3.12 XFA 表单

XFA (XML Forms Architecture) 表单 是基于 XML 的表单，封装在 PDF 内。XFA 提供了基于模板的语法和一系列处理规则，允许用户构建交互式表单。简单的来说，基于模板的语法定义了用户数据的字段。

Foxit PDF SDK 提供了 APIs 用来渲染 XFA 表单、填表、导出和导入表单数据。

备注:

- *Foxit PDF SDK 提供了两个回调类 foxit::addon::xfa::AppProviderCallback 和 foxit::addon::xfa::DocProviderCallback，分别将回调对象通过 Library::RegisterXFAAppProviderCallback 以及 XFADoc 的构造函数注册到 SDK 中。这两个类中的所有函数都是纯虚函数，需要用户自己实现。*
- *使用 XFA form 功能，请确保授权 key 文件中包含 'XFA' 的权限。*

Example:

3.12.1 如何加载 XFADoc 并且显示 XFA 交互式表单

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
```

```
#include "include/pdf/interform/fs_pdfform.h"
#include "include/addon/xfa/fs_xfa.h"

using namespace foxit;
using namespace foxit::common;

using namespace pdf;
using namespace foxit::addon::xfa;

CFS_XFAAppHandler* pXFAAppHandler = new CFS_XFAAppHandler(); // implement from
foxit::addon::xfa::AppProviderCallback
Library::RegisterXFAAppProviderCallback(pXFAAppHandler);
WString input_file = input_path + L"xfa_dynamic.pdf";
PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
        return 1;
}

CFS_XFADocHandler* pXFADocHandler = new CFS_XFADocHandler(); // implement from
foxit::addon::xfa::DocProviderCallback
XFADoc xfa_doc(doc, pXFADocHandler);
xfa_doc.StartLoad(NULL);
...
```

3.12.2 如何导出和导入 XFA 表单数据

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"
#include "include/addon/xfa/fs_xfa.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon::xfa;

// Assuming FSXFADoc xfa_doc has been loaded.

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

xfa_doc.ResetForm();
doc.SaveAs( L"xfa_dynamic_resetform.pdf");

xfa_doc.ImportData(L"xfa_form.xml");
doc.SaveAs(L"xfa_dynamic_importdata.pdf");
...
```

3.13 填表 (Form Filler)

表单填写是用户常用的功能。其允许应用程序动态填写表单。填写表单的关键点是构建一些PDF SDK调用的回调函数。请通过当前 Form对象构建一个 Filler 对象，或者当已经构建该对象则可以通过函数 Form::GetFormFiller 获取 Filler 对象。一个交互式表单只有一个 form filler 对象。对于如何使用 form filler 来填写表单，您可以参考Windows平台包中 "\examples\view_demo" 文件夹下的 "**view_demo**"。

3.14 表单设计 (Form Design)

可填写的 PDF 表单 (AcroForm) 特别适用于各种应用程序表单设计，比如税收和其他政府部门表单。表单设计提供了 APIs 用来向 PDF 文件中添加表单域或者从 PDF 文档中移除表单域。从零开始设计一个表单允许开发人员创建他们需要的内容和布局的表单。

Example:

3.14.1 如何向 PDF 添加一个文本表单域

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.

// Add text field
Control control = form.AddControl(page, L"Text Field0", Field::e_TypeTextField, RectF(50, 600, 90, 640));
control.GetField().SetValue(L"3");
// Update text field's appearance.
control.GetWidget().ResetAppearanceStream();

Control control1 = form.AddControl(page, L"Text Field1", Field::e_TypeTextField, RectF(100, 600, 140, 640));
control1.GetField().SetValue(L"123");
// Update text field's appearance.
control1.GetWidget().ResetAppearanceStream();
...
```

3.14.2 如何从 PDF 中移除一个文本表单域

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/interform/fs_pdfform.h"
```

```
using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
const wchar_t* filter = L"text1";
int countFields = form.GetFieldCount(NULL);
for (int i = 0; i < countFields; i++)
{
    Field field = form.GetField(i, filter);
    if (field.GetType() == Field::e_TypeTextField) {
            form.RemoveField(field);
    }
}
...
```

3.15 注释 (Annotations)

3.15.1 常规注释

一个 annotation 注释将对象（如注释，线条和高亮）与 PDF 文档页面上的位置相关联。其提供了一种通过鼠标和键盘与用户进行交互的方式。PDF 包括如 Table 3-1 中列出的各种标准注释类型。在这些注释类型中，许多被定义为标记注释，因为它们主要用于标记 PDF 文档。标记注释中作为其自身一部分的文本，可以在其他符合标准的阅读器中以其他方式显示，例如在 Comments 面板。Table 3-1 中的 “Markup” 列用来说明是否为标记注释。

Foxit PDF SDK 支持 PDF Reference [1] 中定义的大多数注释类型。Foxit PDF SDK 提供了注释创建，属性访问和修改，外观设置和绘制的 APIs。

Table 3-1

注释类型	描述	Markup	SDK 是否支持
Text(Note)	Text annotation	Yes	Yes
Link	Link Annotation	No	Yes
FreeText (TypeWritter/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

注释类型	描述	Markup	SDK 是否支持
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

备注:

1. Widget 和 watermark 注释类型是比较特殊的。'Annotation' 模块不支持它们。Widget 类型仅在 'form filler' 模块中使用，watermark 类型仅在 'watermark' 模块中使用。
2. Foxit SDK 支持名为 PSI (pressure sensitive ink，压感笔迹) 的自定义注释类型。在 PDF Reference [1]中没有对该注释进行描述。通常，PSI 用于手写签名功能，Foxit SDK 将其视为 PSI 注释，以便其他 PDF 产品可以对其进行相关处理。

Example:

3.15.1.1 如何向 PDF 页面中添加 link 注释

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

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

```
// Assuming the annnots in the page have been loaded.

// Add link annotation.
annots::Link link(page.AddAnnot( Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);
...
```

3.15.1.2 如何向 PDF 页面中添加 highlight 注释，并且设置相关属性

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

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

// Add highlight annotation.
annots::Highlight highlight(page.AddAnnot(Annot::e_Highlight,RectF(10,450,100,550)));
highlight.SetContent(L"Highlight");
annots::QuadPoints quad_points;
quad_points.first = PointF(10, 500);
quad_points.second = PointF(90, 500);
quad_points.third = PointF(10, 480);
quad_points.fourth = PointF(90, 480);
annots::QuadPointsArray quad_points_array;
quad_points_array.Add(quad_points);
highlight.SetQuadPoints(quad_points_array);
highlight.SetSubject(L"Highlight");
highlight.SetTitle(L"Foxit SDK");
highlight.SetCreationDateTime(GetLocalDateTime());
highlight.SetModifiedDateTime(GetLocalDateTime());
highlight.SetUniqueID(WString::FromLocal(RandomUID()));

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

3.15.1.3 如何在创建 markup 注释时设置 popup 信息

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
```

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// 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.
annots::Note note(page.AddAnnot(Annot::e_Note, RectF(10,350,50,400)));
note.SetIconName("Comment");
note.SetSubject(L"Note");
note.SetTitle(L"Foxit SDK");
note.SetContent(L"Note annotation.");
note.SetCreationDateTime(GetLocalDateTime());
note.SetModifiedDateTime(GetLocalDateTime());
note.SetUniqueID(WString::FromLocal(RandomUID()));

// Create a new popup annot and set it to the new note annot.
Popup popup(page.AddAnnot(Annot::e_Popup, RectF(300,450,500,550)));
popup.SetBorderColor(0x00FF00);
popup.SetOpenStatus(false);
popup.SetModifiedDateTime(GetLocalDateTime());
note.SetPopup(popup);
...
```

3.15.1.4 如何使用设备坐标获取 PDF 中特定的注释

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

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

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());

// Get page transformation matrix.
```

```
Matrix displayMatrix= page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());
int iAnnotCount = page.GetAnnotCount();

for(int i=0; i<iAnnotCount; i++)
{
	Annot pAnnot = page.GetAnnot(i);
	ASSERT_FALSE(pAnnot.IsEmpty());
	if (Annot::e_Popup == pAnnot.GetType()) continue;
	RectI annotRect = pAnnot.GetDeviceRect(false, displayMatrix);
	PointF pt;
	float tolerance = 1.0;

	// Get the same annot (pAnnot) using annotRect.
	pt.x = annotRect.left + tolerance;
	pt.y = (annotRect.top - annotRect.bottom)/2 + annotRect.bottom;
	Annot gAnnot = page.GetAnnotAtDevicePoint(pt, tolerance, &displayMatrix);
	...
}
```

3.15.1.5 如何提取 text markup annotation 中的文本内容

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;
...

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

PDFPage page = doc.GetPage(0);
// Parse the first page.
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
int annot_count = page.GetAnnotCount();
TextPage text_page(page);
for (int i = 0; i < annot_count; i++) {
	annots::Annot annot = page.GetAnnot(i);
	annots::TextMarkup text_markup(annot);
	if (!text_markup.IsEmpty()) {
		// Get the texts which intersect with a text markup annotation.
		WString text = text_page.GetTextUnderAnnot(text_markup);
	}
}
```

3.15.1.6 如何为 freetext 注释添加 richtext

```
#include "include/common/fs_common.h"
#include "include/pdf/annots/fs_annot.h"
```

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

// Make sure that SDK has already been initialized successfully.
// Load a PDF document, get a PDF page and parse it.

// Add a new freetext annotation, as text box.
annots::FreeText freetext(pdf_page.AddAnnot(Annot::e_FreeText, RectF(50, 50, 150, 100)));
// Set annotation's properties.

// Add/insert richtext string with style.
RichTextStyle richtext_style;
richtext_style.font = Font(L"Times New Roman", 0, Font::e_CharsetANSI, 0);
richtext_style.text_color = 0xFF0000;
richtext_style.text_size = 10;
freetext.AddRichText(L"Textbox annotation ", richtext_style);

richtext_style.text_color = 0x00FF00;
richtext_style.is_underline = true;
freetext.AddRichText(L"1-underline ", richtext_style);

richtext_style.font = Font(L"Calibri", 0, Font::e_CharsetANSI, 0);
richtext_style.text_color = 0x0000FF;
richtext_style.is_underline = false;
richtext_style.is_strikethrough = true;
int richtext_count = freetext.GetRichTextCount();
freetext.InsertRichText(richtext_count - 1, L"2_strikethrough ", richtext_style);

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

3.15.2 从 FDF 文件导入注释或者将注释导出到 FDF 文件

在 Foxit PDF SDK 中，可以使用来自应用程序或者 FDF 文件的数据来创建注释。同时，PDF SDK 支持将注释导出到 FDF 文件。

Example:

3.15.2.1 如何从 FDF 文件导入注释，并将其添加到 PDF 文档的首页

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace std;
using namespace foxit;
```

```
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace annots;

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

FILE* file = NULL;
#if defined(_WIN32) || defined(_WIN64)
    fopen_s(&file, (const char*)(const char*)String::FromUnicode(fdf_file), "rb+");
#else
    file = fopen((const char*)(const char*)String::FromUnicode(fdf_file), "rb+");
#endif
fseek(file, 0, SEEK_END);
size_t file_size = (size_t)ftell(file);
char* buffer = (char*)malloc(file_size * sizeof(char));
memset(buffer, 0 , file_size);

fseek(file, 0, SEEK_SET);
fread(buffer, sizeof(char), file_size, file);
fclose(file);

fdf::FDFDoc fdf_doc(buffer, file_size);
pdf_doc.ImportFromFDF(fdf_doc, PDFDoc::e_Annots);
```

3.16 图片转换 (Image Conversion)

Foxit PDF SDK 提供了 PDF 文件和图片之间进行转换的 APIs. 应用程序可以轻松地实现图片创建和图片转换等功能，支持如下的图片格式：BMP、TIFF、PNG、JPX、JPEG 和 GIF。通过 Foxit PDF SDK，PDF 文件和支持的图片格式 (除了 GIF) 之间可以互相转换。Foxit PDF SDK 只支持将 GIF 图片转换为 PDF 文件。

Example:

3.16.1 如何将 PDF 页面转换为位图文件

```
#include "include/common/fs_common.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;

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

```
// Get page count.
int nPageCount = doc.GetPageCount();
for(int i=0;i<nPageCount;i++) {
   PDFPage page = doc.GetPage(i);

   // Parse page.
   page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

   int width = static_cast<int>(page.GetWidth());
   int height = static_cast<int>(page.GetHeight());
   Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

   // Prepare a bitmap for rendering.
   Bitmap bitmap(width, height, foxit::common::Bitmap::e_DIBArgb, NULL, 0);
   bitmap.FillRect(0xFFFFFFFF, NULL);

   // Render page.
   Renderer render(bitmap, false);
   render.StartRender(page, matrix, NULL);
      image.AddFrame(bitmap);
}
...
```

备注：对于pdf2image功能，如果PDF文件中包含了 大于1G的图片，建议使用分块渲染处理图片。否则，可能会出现异常。以下是分块渲染的简要实现。

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;

...
page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null, false);
int render_sum = 10;
int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());

int width_scale = 1;
int height_scale = 1;
int little_width = width * width_scale;
int little_height = height / render_sum * height_scale;

for (size_t i = 0; i<render_sum; i++)
{
   // According to Matrix, do module rendering for large PDF files.
   Matrix matrix = page.GetDisplayMatrix(0, -1 * i * little_height, width * width_scale, height * height_scale,
e_Rotation0);
   // Prepare a bitmap for rendering.
   Bitmap bitmap(little_width, little_height, Bitmap::e_DIBArgb, NULL, 0);
```

```
    bitmap.FillRect(0xFFFFFFFF, NULL);
    Renderer render(bitmap, false);
    render.StartRender(page, matrix, NULL);
    // The bitmap data will be added to the end of image file after rendering.
    ...
}
```

3.16.2 如何将图片转换为 PDF 文件

```
#include "include/common/fs_common.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;

Image image(input_file);
int count = image.GetFrameCount();

PDFDoc doc;
for (int i = 0; i < count; i++) {
    PDFPage page = doc.InsertPage(i);
    page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
    // Add image to page.
    page.AddImage(image, i, PointF(0, 0), page.GetWidth(), page.GetHeight(), true);
}

doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.17 水印 (Watermark)

水印是一种 PDF 注释，广泛用于 PDF 文档。水印是文档上嵌入的可见叠加层，包含文本、logo 或版权声明。水印的目的是对作者工作成果的保护，防止其未经授权而被他人使用。Foxit PDF SDK 提供了允许应用程序创建、插入和删除水印的 APIs。

Example:

3.17.1 如何创建一个文本水印，并将其插入到 PDF 文档的第一页

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_watermark.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
```

```
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

WatermarkSettings settings;
settings.flags = WatermarkSettings::e_FlagASPageContents | WatermarkSettings::e_FlagOnTop;
settings.offset_x = 0;
settings.offset_y = 0;
settings.opacity = 90;
settings.position = common::e_PosTopRight;
settings.rotation = -45.f;
settings.scale_x = 1.f;
settings.scale_y = 1.f;

WatermarkTextProperties text_properties;
text_properties.alignment = CommonDefines::e_AlignmentCenter;
text_properties.color = 0xF68C21;
text_properties.font_style = WatermarkTextProperties::e_FontStyleNormal;
text_properties.line_space = 1;
text_properties.font_size = 12.f;
text_properties.font = Font(Font::e_StdIDTimesB);

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

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

3.17.2 如何创建一个图片水印，并将其插入到 PDF 文档的第一页

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_watermark.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

WatermarkSettings settings;
settings.flags = WatermarkSettings::e_FlagASPageContents | WatermarkSettings::e_FlagOnTop;
settings.offset_x = 0.f;
settings.offset_y = 0.f;
settings.opacity = 20;
settings.position = common::e_PosCenter;
settings.rotation = 0.0f;
```

```
Image image(image_file);
Bitmap bitmap = image.GetFrameBitmap(0);
settings.scale_x = page.GetWidth() * 0.618f / bitmap.GetWidth();
settings.scale_y = settings.scale_x;

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

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

3.17.3 如何从 PDF 页面中删除所有的水印

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.
...
page.RemoveAllWatermarks();
...
// Save document to file
...
```

3.18 条形码 (Barcode)

条形码用于表示与某个对象相关的数据，该数据可通过光学机器进行读取。最初的条形码系统是通过平行线间宽度和间距的不同来表示数据，可称为线性条形码或一维条形码(1D)。后来条形码逐渐演变成矩形、点、六边形等 2D 几何图案。虽然 2D 系统使用了一系列符号，但是它们通常也被称为条形码。条形码最初由特定的光学扫描器进行扫描，该光学扫面器被称为条形码读取器。后来扫描器和解释软件成功应用于桌面打印机和智能手机等设备。Foxit PDF SDK 提供了从给定字符串生成条形码位图的应用程序。Table 3-2 列出了 Foxit PDF SDK 支持的条形码类型。

Table 3-2

条形码类型	Code39	Code128	EAN8	UPCA	EAN13	ITF	PDF417	QR
维度	1D	1D	1D	1D	1D	1D	2D	2D

Example:

3.18.1 如何从字符串生成条形码位图

```
#include "include/common/fs_common.h"
```

```
#include "include/common/fs_barcode.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
...

// Strings used as barcode content.
WString sz_code_string = L"TEST-SHEET";

// Barcode format types.
Barcode::Format code_format = Barcode::e_FormatCode39;

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

//Image names for the saved image files for QR code.
WString bmp_qr_name = L"/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;
Bitmap bitmap = barcode.GenerateBitmap(sz_code_string, code_format, unit_width, unit_height, sz_qr_level);
...
```

3.19 安全 (Security)

Foxit PDF SDK 提供了一系列加密和解密功能，以满足不同级别的文档安全保护。用户可以使用常规密码加密和证书驱动加密，或使用自己的安全处理机制来自定义安全实现。另外，Foxit PDF SDK 还提供了 APIs 用于集成第三方安全技术 (Microsoft RMS)，允许开发人员使用 Microsoft RMS SDK 加密和解密 PDF 文档。

备注：*有关RMS加密和解密更详细的信息，请参考SDK包中 "\examples\simple_demo" 文件夹下的 "**security**" demo。*

Example:

3.19.1 如何使用证书加密 PDF 文件

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_security.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
```

```
...
PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
   if (error_code != foxit::e_ErrSuccess) {
     return false;
   }

// Do encryption.
StringArray envelopes;
String initial_key;
WString cert_file_path = input_path + L"foxit.cer";
if (!GetCertificateInfo((const char*)String::FromUnicode(cert_file_path), envelopes, initial_key, true, 16)) {
    return false;
}
CertificateSecurityHandler handler;
CertificateEncryptData encrypt_data(true, SecurityHandler::e_CipherAES, envelopes);
handler.Initialize(encrypt_data, initial_key);

doc.SetSecurityHandler(handler);
WString output_file = output_directory + L"certificate_encrypt.pdf";
doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.19.2 如何使用 Foxit DRM 加密 PDF 文件

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_security.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
  return false;
}

// Do encryption.
DRMSecurityHandler handler = DRMSecurityHandler();
const char* file_id = "Simple-DRM-file-ID";
String initialize_key = "Simple-DRM-initialize-key";
DRMEncryptData encrypt_data(true, "Simple-DRM-filter", SecurityHandler::e_CipherAES, 16, true, 0xfffffffc);
handler.Initialize(encrypt_data, file_id, initialize_key);
doc.SetSecurityHandler(handler);

WString output_file = output_directory + L"foxit_drm_encrypt.pdf";
doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.20 页面重排 (Reflow)

页面重排功能是在页面大小发生变化时自动重排页面内容。该功能对那些需要在不同尺寸的输出设备上显示 PDF 文档的应用程序具有很大的利用价值。页面重排让应用程序无需考虑设备的不同尺寸。Foxit PDF SDK 提供了 APIs 用来创建、渲染、释放 Reflow 页面，以及访问 Reflow 页面的属性。

Example:

3.20.1 如何创建一个 Reflow 页面，并将其渲染为位图文件

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_reflowpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

PDFPage page = doc.GetPage(0);
// Parse PDF page.
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

ReflowPage reflow_page(page);

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

// 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(bitmap, false);
foxit::Matrix matrix = reflow_page.GetDisplayMatrix(0, 0);
renderer.StartRenderReflowPage(reflow_page, matrix, NULL);
...
```

3.21 异步加载 PDF (Asynchronous PDF)

异步加载 PDF 技术是一种在文档加载需要花费很长时间时，可以不用加载整个文档就可以对 PDF 页面进行访问的方法。该方法专为访问互联网上的 PDF 文件而设计。使用异步 PDF 技术，应用程序无需等待下载整个 PDF 文件就可以对其进行访问，可以打开任何已经下载加载的 PDF 页面。该技术为 Web 阅读类的应用程序提供了一种方便和有效的方式。关于如何使用异步模式打开和解析 PDF 页面，请参考 SDK 包中 "\examples\simple_demo" 文件夹下的 "**async_load**" demo。

3.22 压感笔迹 (Pressure Sensitive Ink)

压感笔迹 (PSI) 是一种获取变化电力输出以响应作用于压力感应设备元件上的各种变化压力或受力的技术。在 PDF 中，PSI 通常被用于手写签名，通过捕捉手指或触控笔的压力变化来收集 PSI 数据。PSI 数据包含操作区域的坐标和画布，并用其来绘制 PSI 的外观。Foxit PDF SDK 允许应用程序创建 PSI、访问其属性、操作 ink 笔迹和画布、以及释放 PSI。

Example:

3.22.1 如何创建 PSI 并设置相关属性

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_psi.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace annots;

PSI 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::e_TypeMoveTo;
psi.AddPoint(PointF(x, y), type, pressure);
...
```

3.23 Wrapper

Wrapper 为用户提供了一种保存与 PDF 文档相关的数据的方法。例如，在打开一个加密未授权的 PDF 文档，用户会看到错误信息提示其没有权限访问该文档。在这种情况下，使用 wrapper，用户即使无法访问 PDF 中的内容，但仍然可以访问该文档的 wrapper 数据。Wrapper 数据可用来提供信息给用户，比如文档的解密方法。

Example:

3.23.1 如何打开包含 wrapper 数据的 PDF 文档

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace common::file;

// file_name is PDF document which includes wrapper data.
PDFDoc doc(file_name);
ErrorCode code = doc.Load();
if (code != foxit::e_ErrSuccess) {
   return false;
}
if(!doc.IsWrapper()){
   return false;
}
int64 offset = doc.GetWrapperOffset();

FileReader file_reader(offset);
file_reader.LoadFile(String::FromUnicode(file_name));
...
```

3.24 PDF 对象 (PDF Objects)

PDF 中有八种类型的对象：布尔对象、数字对象、字符串对象、名称对象、数组对象、字典对象、流对象和空对象。 PDF 对象是文档级文档，与页面对象（见 3.24）不同，每个页面对象都与特定的页面相关联。Foxit PDF SDK 提供了 APIs 用来创建、修改、检索和删除文档中的这些对象。

Example:

3.24.1 如何从目录字典中删除指定的属性

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
```

```
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
...

// Assuming PDFDoc doc has been loaded.

PDFDictionary* catalog = doc.GetCatalog();
if (NULL == catalog) return;

const char* key_strings[] = { "Type", "Boolean", "Name", "String", "Array", "Dict"};
int count = sizeof(key_strings)/sizeof(key_strings[0]);
for (int i = 0; i < count; i ++) {
  if (catalog->HasKey(key_strings[i]))
     catalog->RemoveAt(key_strings[i]);
}
...
```

3.25 页面对象 (Page Object)

页面对象可以帮助对 PDF 对象 (关于 PDF 对象更详细的介绍，见 3.23) 知识了解有限的开发人员能够处理 PDF 文档中的文本、路径、图像和画布等对象。Foxit PDF SDK 提供 APIs 用以在页面中添加和删除 PDF 对象并设置特定属性。使用页面对象，用户可以从对象内容创建 PDF 页面。页面对象的其他可能用法包括向 PDF 文档添加页眉和页脚，向每个页面添加图片 logo，或者根据需要生成 PDF 模板。

Example:

3.25.1 如何在 PDF 页面中创建一个文本对象

```
#include "include/common/fs_common.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...

// Assuming PDFPage page has been loaded and parsed.

POSITION position = page.GetLastGraphicsObjectPosition(GraphicsObject::e_TypeText);
TextObject* text_object = TextObject::Create();
```

```
text_object->SetFillColor(0xFFFF7F00);

// Prepare text state.
TextState state;
state.font_size = 80.0f;
state.font =  Font(L"Simsun", Font::e_StylesSmallCap, Font::e_CharsetGB2312, 0);
state.textmode = TextState::e_ModeFill;
text_object->SetTextState(page, state, false, 750);

// Set text.
text_object->SetText(L"Foxit Software");
POSITION last_position = page.InsertGraphicsObject(position, text_object);
...
```

3.25.2 如何向 PDF 页面中插入一个图片 logo

```
#include "include/common/fs_common.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...
//Assuming PDFPage page has been loaded and parsed.

POSITION position = page.GetLastGraphicsObjectPosition(GraphicsObject::e_TypeImage);
Image image(image_file);
ImageObject* image_object = ImageObject::Create(page.GetDocument());
image_object->SetImage(image, 0);

float width = static_cast<float>(image.GetWidth());
float height = static_cast<float>(image.GetHeight());

float page_width = page.GetWidth();
float page_height = page.GetHeight();

// Please notice the matrix value.
image_object->SetMatrix(Matrix(width, 0, 0, height, (page_width - width) / 2.0f, (page_height - height) / 2.0f));

page.InsertGraphicsObject(position, image_object);
page.GenerateContent();
...
```

3.26 标记内容 (Marked content)

在 PDF 文档中，可以将一部分内容标记为标记内容元素。标记内容功能有助于管理 PDF 文档的逻辑结构信息并且可以用于生成加标记的 PDF (tagged PDF) 。加标记的 PDF 具有标准的结构类型和属性，有助于提取和再利用页面内容。有关标记内容的更多详细信息，请参阅 PDF reference 1.7 [1] 的第 10.5 章。

Example:

3.26.1 如何获取页面中的标记内容以及 tag 名称

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_image.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
using namespace objects;

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

POSITION position = page.GetFirstGraphicsObjectPosition(GraphicsObject::e_TypeText);
TextObject* text_obj = reinterpret_cast<TextObject*>(page.GetGraphicsObject(position));
MarkedContent* content = text_obj->GetMarkedContent();
int item_count = content->GetItemCount();

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

3.27 PDF 图层 (PDF Layer)

Foxit PDF SDK 支持 PDF 图层，也称为可选内容组 (Optional Content Groups，OCG) 。用户可以选择性地查看或隐藏多图层 PDF 文档的不同层中的内容。多图层广泛用于许多应用领域，如 CAD 制图、地图、分层艺术品以及多语言文档等。

在 Foxit PDF SDK 中，PDF 图层与图层节点相关联。要获取图层节点，用户应首先构建 PDF LayerTree 对象，然后调用函数 LayerTree::GetRootNode 以获取整个图层树的根图层节点。另外，

您可以从根图层节点开始枚举图层树中的所有节点。Foxit PDF SDK 提供 APIs 用来获取/设置图层数据，查看或隐藏不同图层中的内容，设置图层名称，添加或删除图层，以及编辑图层。

Example:

3.27.1 如何创建一个 PDF 图层

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace pdf;
...
// Assuming PDFDoc doc has been loaded.

LayerTree layertree(doc);
LayerNode root = layertree.GetRootNode();
if (root.IsEmpty()) {
        printf("No layer information!\r\n");
        return ;
}
...
```

3.27.2 如何设置所有图层节点的信息

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
 ...
// Assuming PDFDoc doc has been loaded.

void SetAllLayerNodesInformation(LayerNode layer_node) {
  if (layer_node.HasLayer()) {
    layer_node.SetDefaultVisible(true);
    layer_node.SetExportUsage(LayerTree::e_StateUndefined);
    layer_node.SetViewUsage(LayerTree::e_StateOFF);
    LayerPrintData print_data("subtype_print", LayerTree::e_StateON);
    layer_node.SetPrintUsage(print_data);
    LayerZoomData zoom_data(1, 10);
    layer_node.SetZoomUsage(zoom_data);
    WString new_name = WString(L"[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++) {
    LayerNode child = layer_node.GetChild(i);
    SetAllLayerNodesInformation(child);
  }
}

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

3.27.3 如何编辑图层树

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
 ...
// Assuming PDFDoc doc has been loaded.

// edit layer tree
PDFDoc doc(input_file);
error_code = doc.Load();
layertree = LayerTree(doc);
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, L"AddedLayerNode", true);
child.AddChild(0, L"AddedNode", false);
```

3.28 签名 (Signature)

PDF 签名可用于创建和签署 PDF 文档的数字签名，从而保护文档内容的安全性并避免文档被恶意篡改。它可以让接收者确保其收到的文档是由签名者发送的，并且文档内容是完整和未被经篡的。Foxit PDF SDK 提供 APIs 用来创建数字签名，验证签名的有效性，删除现有的数字签名，获取和设置数字签名的属性，显示签名和自定义签名表单域的外观。

备注：*Foxit PDF SDK 提供了默认签名回调函数，支持如下两种类型的 signature filter 和 subfilter:*

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

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

如果您使用以上任意一种的 signature filter 和 subfilter，您可以直接签名 PDF 文档和验证签名的有效性，而不需要注册自定义回调函数。

Example:

3.28.1 如何对 PDF 文档进行签名

```
#include "include/pdf/annots/fs_annot.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
using namespace file;

// AdobePPKLiteSignature
const char* filter = "Adobe.PPKLite";
const char* sub_filter = "adbe.pkcs7.detached";

if (!use_default) {
	InitializeOpenssl();
	sub_filter = "adbe.pkcs7.sha1";
	SignatureCallbackImpl* sig_callback = new SignatureCallbackImpl(sub_filter);
	Library::RegisterSignatureCallback(filter, sub_filter, sig_callback);
}

printf("Use signature callback object for filter \"%s\" and sub-filter \"%s\"\r\n",
	filter, sub_filter);
PDFPage pdf_page = pdf_doc.GetPage(0);
// Add a new signature to the first page.
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);
bool is_signed = new_signature.IsSigned();
uint32 sig_state = new_signature.GetState();
printf("[Before signing] Signed?:%s\t State:%s\r\n",
	is_signed? "true" : "false",
	TransformSignatureStateToString(sig_state).c_str());

// Sign the new signature.
WString signed_pdf_path = output_directory + L"signed_newsignature.pdf";
if (use_default)
	signed_pdf_path = output_directory + L"signed_newsignature_default_handler.pdf";

WString cert_file_path = input_path + L"foxit_all.pfx";
WString cert_file_password = L"123456";
```

```
// Cert file path will be passed back to application through callback function FSSignatureCallback::Sign().
// In this demo, the cert file path will be used for signing in callback function FSSignatureCallback::Sign().
new_signature.StartSign((const wchar_t*)cert_file_path, cert_file_password,
            Signature::e_DigestSHA1, (const wchar_t*)signed_pdf_path, NULL, NULL);
printf("[Sign] Finished!\r\n");
is_signed = new_signature.IsSigned();
sig_state = new_signature.GetState();
printf("[After signing] Signed?:%s\tState:%s\r\n",
            is_signed? "true" : "false",
            TransformSignatureStateToString(sig_state).c_str());

// Open the signed document and verify the newly added signature (which is the last one).
printf("Signed PDF file: %s\r\n", (const char*)String::FromUnicode(signed_pdf_path));
PDFDoc signed_pdf_doc((const wchar_t*)signed_pdf_path);
ErrorCode error_code = signed_pdf_doc.Load(NULL);
if (foxit::e_ErrSuccess !=error_code ) {
        printf("Fail to open the signed PDF file.\r\n");
        return;
}
// Get the last signature which is just added and signed.
int sig_count = signed_pdf_doc.GetSignatureCount();
Signature signed_signature = signed_pdf_doc.GetSignature(sig_count-1);
// Verify the signature.
signed_signature.StartVerify(NULL, NULL);
printf("[Verify] Finished!\r\n");
is_signed = signed_signature.IsSigned();
sig_state = signed_signature.GetState();
printf("[After verifying] Signed?:%s\tState:%s\r\n",
            is_signed? "true" : "false",
            TransformSignatureStateToString(sig_state).c_str());
```

3.28.2 如何实现签名的回调函数

```
#include "include/pdf/annots/fs_annot.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
using namespace file;

// Implementation of pdf::SignatureCallback
class SignatureCallbackImpl : public pdf::SignatureCallback {
public:
        SignatureCallbackImpl(string subfilter)
                : sub_filter_(subfilter)
                , digest_context_(NULL) {}
```

```
        ~SignatureCallbackImpl();

        virtual void Release() {
                delete this;
        }
        virtual bool StartCalcDigest(const ReaderCallback* file, const uint32* byte_range_array,
                uint32 size_of_array, const Signature& signature, const void* client_data);
        virtual Progressive::State ContinueCalcDigest(const void* client_data, const PauseCallback* pause);
        virtual String GetDigest(const void* client_data);
        virtual String Sign(const void* digest, uint32 digest_length, const wchar_t* cert_path,
                const WString& password, Signature::DigestAlgorithm digest_algorithm,
                void* client_data);
        virtual uint32 VerifySigState(const void* digest, uint32 digest_length,
                const void* signed_data, uint32 signed_data_len,
                void* client_data);
        virtual bool IsNeedPadData() {return false;}
protected:
        bool GetTextFromFile(unsigned char *plainString);

        unsigned char* PKCS7Sign(const wchar_t* cert_file_path, String cert_file_password,
                String plain_text, int& signed_data_size);
        bool PKCS7VerifySignature(String signed_data, String plain_text);
        bool ParseP12File(const wchar_t* cert_file_path, String cert_file_password,
                EVP_PKEY** pkey, X509** x509, STACK_OF(X509)** ca);
        ASN1_INTEGER* CreateNonce(int bits);

private:
        string sub_filter_;
        DigestContext* digest_context_;

        string cert_file_path_;
        string cert_file_password_;
};

SignatureCallbackImpl::~SignatureCallbackImpl() {
        if (digest_context_) {
                delete digest_context_;
                digest_context_ = NULL;
        }
}

bool SignatureCallbackImpl::GetTextFromFile(unsigned char* file_buffer) {
        if (!digest_context_ || !digest_context_->GetFileReadCallback()) return false;
        ReaderCallback* file_read = digest_context_->GetFileReadCallback();
        file_read->ReadBlock(file_buffer, digest_context_->GetByteRangeElement(0),
digest_context_->GetByteRangeElement(1));
        file_read->ReadBlock(file_buffer + (digest_context_->GetByteRangeElement(1)-
digest_context_->GetByteRangeElement(0)),
                digest_context_->GetByteRangeElement(2), digest_context_->GetByteRangeElement(3));
        return true;
}
```

```
bool SignatureCallbackImpl::StartCalcDigest(const ReaderCallback* file, const uint32* byte_range_array,
        uint32 size_of_array, const Signature& signature, const void* client_data) {
                if (digest_context_) {
                        delete digest_context_;
                        digest_context_ = NULL;
                }
                digest_context_ = new DigestContext(const_cast<ReaderCallback*>(file), byte_range_array,
size_of_array);
                if(!SHA1_Init(&digest_context_->sha_ctx_)) {
                        delete digest_context_;
                        digest_context_ = NULL;
                        return false;
                }
                return true;
}

Progressive::State SignatureCallbackImpl::ContinueCalcDigest(const void* client_data, const PauseCallback*
pause) {
        if (!digest_context_) return Progressive::e_Error;

        uint32 file_length = digest_context_->GetByteRangeElement(1) +
digest_context_->GetByteRangeElement(3);
        unsigned char* file_buffer = (unsigned char*)malloc(file_length);
        if (!file_buffer || !GetTextFromFile(file_buffer)) return Progressive::e_Error;

        SHA1_Update(&digest_context_->sha_ctx_, file_buffer, file_length);
        free(file_buffer);
        return Progressive::e_Finished;
}

String SignatureCallbackImpl::GetDigest(const void* client_data) {
        if (!digest_context_) return "";
        unsigned char* md = reinterpret_cast<unsigned
char*>(OPENSSL_malloc((SHA_DIGEST_LENGTH)*sizeof(unsigned char)));
        if (1 != SHA1_Final(md, &digest_context_->sha_ctx_))
                return "";
        String digest = String(reinterpret_cast<const char*>(md), SHA_DIGEST_LENGTH);
        OPENSSL_free(md);
        return digest;
}

String SignatureCallbackImpl::Sign(const void* digest, uint32 digest_length, const wchar_t* cert_path,
        const WString& password, Signature::DigestAlgorithm digest_algorithm,
        void* client_data) {
                if (!digest_context_) return "";
                String plain_text;
                if ("adbe.pkcs7.sha1" == sub_filter_) {
                        plain_text = String((const char*)digest, digest_length);
                }
                int signed_data_length = 0;
                unsigned char* signed_data_buffer = PKCS7Sign(cert_path, String::FromUnicode(password),
                        plain_text, signed_data_length);
```

```
                if (!signed_data_buffer) return "";

                String signed_data = String((const char*)signed_data_buffer, signed_data_length);
                free(signed_data_buffer);
                return signed_data;
}

uint32 SignatureCallbackImpl::VerifySigState(const void* digest, uint32 digest_length,
        const void* signed_data, uint32 signed_data_len, void* client_data) {
                // Usually, the content of a signature field is contain the certification of signer.
                // But we can't judge this certification is trusted.
                // For this example, the signer is ourself. So when using api PKCS7_verify to verify,
                // we pass NULL to it's parameter <i>certs</i>.
                // Meanwhile, if application should specify the certificates, we suggest pass flag
PKCS7_NOINTERN to
                // api PKCS7_verify.
                if (!digest_context_) return Signature::e_StateVerifyErrorData;
                String plain_text;
                unsigned char* file_buffer = NULL;
                if ("adbe.pkcs7.sha1" == sub_filter_) {
                        plain_text = String(reinterpret_cast<const char*>(digest), digest_length);
                } else {
                        return Signature::e_StateUnknown;
                }

                String signed_data_str = String(reinterpret_cast<const char*>(signed_data), signed_data_len);
                bool ret = PKCS7VerifySignature(signed_data_str, plain_text);
                if (file_buffer) free(file_buffer);
                return ret ? Signature::e_StateVerifyNoChange : Signature::e_StateVerifyChange;

}

ASN1_INTEGER* SignatureCallbackImpl::CreateNonce(int bits) {
        unsigned char buf[20];
        int len = (bits - 1) / 8 + 1;
        // Generating random byte sequence.
        if (len > (int)sizeof(buf)) {
                return NULL;
        }
        if (RAND_bytes(buf, len) <= 0) {
                return NULL;
        }
        // Find the first non-zero byte and creating ASN1_INTEGER object.
        int i = 0;
        for (i = 0; i < len && !buf[i]; ++i) ;
        ASN1_INTEGER* nonce = NULL;
        if (!(nonce = ASN1_INTEGER_new())) {
                ASN1_INTEGER_free(nonce);
                return NULL;
        }
        OPENSSL_free(nonce->data);
        // Allocate at least one byte.
```

```
        nonce->length = len - i;
        if (!(nonce->data = reinterpret_cast<unsigned char*>(OPENSSL_malloc(nonce->length + 1)))) {
                ASN1_INTEGER_free(nonce);
                return NULL;
        }
        memcpy(nonce->data, buf + i, nonce->length);
        return nonce;
}

bool SignatureCallbackImpl::ParseP12File(const wchar_t* cert_file_path, String cert_file_password,
        EVP_PKEY** pkey, X509** x509, STACK_OF(X509)** ca) {
                FILE* file = NULL;
#if defined(_WIN32) || defined(_WIN64)
                _wfopen_s(&file, cert_file_path, L"rb");
#else
                file = fopen(String::FromUnicode(cert_file_path), "rb");
#endif  // defined(_WIN32) || defined(_WIN64)
                if (!file) {
                        return false;
                }

                PKCS12* pkcs12 = d2i_PKCS12_fp(file, NULL);
                fclose (file);
                if (!pkcs12) {
                        return false;
                }

                if (!PKCS12_parse(pkcs12, (const char*)cert_file_password, pkey, x509, ca)) {
                        return false;
                }

                PKCS12_free(pkcs12);
                if (!pkey)
                        return false;
                return true;
}

unsigned char* SignatureCallbackImpl::PKCS7Sign(const wchar_t* cert_file_path, String cert_file_password,
        String plain_text, int& signed_data_size) {
                PKCS7* p7 = NULL;
                EVP_PKEY* pkey = NULL;
                X509* x509 = NULL;
                STACK_OF(X509)* ca = NULL;
                if(!ParseP12File(cert_file_path, cert_file_password, &pkey, &x509, &ca))
                        return NULL;

                p7 = PKCS7_new();
                PKCS7_set_type(p7, NID_pkcs7_signed);
                PKCS7_content_new(p7, NID_pkcs7_data);

                // Application should not judge the sign algorithm with the content's length.
                // Here, just for convenient;
```

```
                if (plain_text.GetLength() > 32)
                        PKCS7_ctrl(p7, PKCS7_OP_SET_DETACHED_SIGNATURE, 1, NULL);

                PKCS7_SIGNER_INFO* signer_info = PKCS7_add_signature(p7, x509, pkey, EVP_sha1());
                PKCS7_add_certificate(p7, x509);

                for (int i = 0; i< sk_num(CHECKED_STACK_OF(X509,ca)); i++)
                        PKCS7_add_certificate(p7, (X509*)sk_value(CHECKED_STACK_OF(X509,ca), i));

                // Set source data to BIO.
                BIO* p7bio = PKCS7_dataInit(p7, NULL);
                BIO_write(p7bio, plain_text.GetBuffer(1), plain_text.GetLength());
                PKCS7_dataFinal(p7, p7bio);

                FREE_CERT_KEY;
                BIO_free_all(p7bio);
                // Get signed data.
                unsigned long der_length = i2d_PKCS7(p7, NULL);
                unsigned char* der = reinterpret_cast<unsigned char*>(malloc(der_length));
                memset(der, 0, der_length);
                unsigned char* der_temp = der;
                i2d_PKCS7(p7, &der_temp);
                PKCS7_free(p7);
                signed_data_size = der_length;
                return (unsigned char*)der;
}

bool SignatureCallbackImpl::PKCS7VerifySignature(String signed_data, String plain_text) {
        // Retain PKCS7 object from signed data.
        BIO* vin = BIO_new_mem_buf((void*)signed_data.GetBuffer(1), signed_data.GetLength());
        PKCS7* p7 = d2i_PKCS7_bio(vin, NULL);
        STACK_OF(PKCS7_SIGNER_INFO) *sk = PKCS7_get_signer_info(p7);
        int sign_count = sk_PKCS7_SIGNER_INFO_num(sk);

        int length = 0;
        bool bSigAppr = false;
        unsigned char *p = NULL;
        for(int i=0;i<sign_count; i++) {
                PKCS7_SIGNER_INFO* sign_info = sk_PKCS7_SIGNER_INFO_value(sk,i);

                BIO *p7bio = BIO_new_mem_buf((void*)plain_text.GetBuffer(1), plain_text.GetLength());
                X509 *x509= PKCS7_cert_from_signer_info(p7,sign_info);
                if(1 == PKCS7_verify(p7, NULL, NULL,p7bio, NULL, PKCS7_NOVERIFY))
                        bSigAppr = true;
                BIO_free(p7bio);
        }
        PKCS7_free(p7);
        BIO_free(vin);
        return bSigAppr;
}
```

3.29 长期签名验证(LTV，Long term validation)

从 7.0 版本开始，Foxit PDF SDK 提供了 API 接口进行长期签名验证，主要用于解决已经过期的签名的验证问题。LTV 需要 DSS(Document Security Store)，其包含了签名的验证信息，以及需要文档时间戳签名 DTS (Document Timestamp Signature)，其是 time stamp 类型的 signature。

为了支持 LTV，Foxit PDF SDK 提供了：

- 支持添加 time stamp 类型的 signature，以及提供了 sub filter "ETSI.RFC3161" 的默认签名回调。
- TimeStampServerMgr 和 TimeStampServer 类，用于 time stamp 的 server 设置和管理等。sub filter "ETSI.RFC3161" 的默认签名回调将会使用默认的 time stamp server。
- LTVVerifier 类，其提供了验证签名和向文档中添加 DSS 信息的功能。同时，也提供了 LTVVerifier 所需的一个基本默认的回调函数 RevocationCallback。

以下仅以使用 SDK 默认的 sub filter "ETSI.RFC3161" 签名回调及默认的 RevocationCallback 为例来说明如何进行长期签名验证。有关更详细的信息，请参阅下载包中 "\examples\simple_demo"目录下的 "**ltv**" demo。

Example:

3.29.1 如何使用 SDK 默认的 sub filter "ETSI.RFC3161" 签名回调及默认的 RevocationCallback 进行长期签名验证

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"
#include "include/pdf/fs_ltvverifier.h"
...

// 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();
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.
PDFDoc pdf_doc(signed_pdf_path);
pdf_doc.StartLoad();
{
	// Use LTVVerifier to verify and add DSS.
	LTVVerifier ltv_verifier(pdf_doc, true, true, false, LTVVerifier::e_SignatureCreationTime);
	// Set verifying mode which is necessary.
	ltv_verifier.SetVerifyMode(LTVVerifier::e_VerifyModeAcrobat);
	SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
	for (size_t 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() & Signature::e_StateVerifyValid)
                            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.
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(RectF(), L"", Signature::e_SignatureTypeTimeStamp);
Progressive sign_progressive = timestamp_signature.StartSign(L"", L"", Signature::e_DigestSHA256,
saved_ltv_pdf_path);
if (sign_progressive.GetRateOfProgress() != 100)
            sign_progressive.Continue();

// Then use LTVVeirfier to verify the new signed PDF file.
PDFDoc check_pdf_doc(saved_ltv_pdf_path);
check_pdf_doc.StartLoad();
{
            // Use LTVVeirfier to verify.
            LTVVerifier ltv_verifier(pdf_doc, true, true, false, LTVVerifier::e_SignatureCreationTime);
            // Set verifying mode which is necessary.
            ltv_verifier.SetVerifyMode(LTVVerifier::e_VerifyModeAcrobat);
            SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
            for (size_t 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();
```

3.30 PAdES

从 7.0 版本开始，Foxit PDF SDK 支持 PAdES (PDF Advanced Electronic Signature)，其是 CAdES 签名在 PDF 中的应用。CAdES 是高级数字签名的一种新标准，其默认 subfilter 是 "ETSI.CAdES.detached"。PAdES 签名分为四个等级：B-B, B-T, B-LT, 和 B-LTA。

- B-B: 包含基本的必须出现的属性。
- B-T: 在 B-B 的基础上，包含文档时间戳或者签名时间戳，来为存在的签名提供可信的时间。
- B-LT: 在 B-T 的基础上，包含 DSS/VRI，来提供证书和吊销信息。
- B-LTA: 在 B-LT 的基础上，为存在的吊销信息提供可信时间 DTS。

Foxit PDF SDK 提供了 subfilter 为 "ETSI.CAdES.detached" 的默认签名回调，可用来签名和验证 subfilter 为 "ETSI.CAdES.detached" 的签名。还提供了 TimeStampServerMgr 和 TimeStampServer 类，用于设置和管理 time stamp server。subfilter "ETSI.CAdES.detached" 的默认签名回调将会使用默认的 time stamp server。

Foxit PDF SDK 提供了从签名中获取 PAdES 不同等级的方法，应用层面也可以根据各个等级的要求来判定所属等级。有关如何在 PDF 文档中添加、签名、和验证 PAdES 签名的更详细信息，请参阅下载包中 "\examples\simple_demo"目录下的 "**pades**" demo。

3.31 PDF 行为 (PDF Action)

PDF Action 代表 PDF 操作类的基类。Foxit PDF SDK 提供了 APIs 用来创建一系列行为，并获取行为句柄，比如 embedded goto action, JavaScript action, named action 和 launch action 等。

Example:

3.31.1 如何创建一个 URI 行为并将其插入到 link 注释

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

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

// Add link annotation
annots::Link link(page.AddAnnot(Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);
// Add action for link annotation
using foxit::pdf::actions::Action;
using foxit::pdf::actions::URIAction;
URIAction action = (URIAction)Action::Create(page.GetDocument(), Action::e_TypeURI);
action.SetTrackPositionFlag(true);
action.SetURI("www.foxitsoftware.com");
link.SetAction(action);
// Appearance should be reset.
link.ResetAppearanceStream();
```

3.31.2 如何创建一个 GoTo 行为并将其插入到 link 注释

```
#include "include/common/fs_common.h"
```

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

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

// Add link annotation
annots::Link link(page.AddAnnot(Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);

GotoAction action = Action::Create(page.GetDocument(), Action::e_TypeGoto);
Destination newDest = Destination::CreateXYZ(page.GetDocument(), 0,0,0,0);
action.SetDestination(newDest);
```

3.32 JavaScript

创建 JavaScript 是为了将 Web 页面的相关处理从服务器转移到基于 Web 的应用程序的客户端上。Foxit PDF SDK JavaScript 以 JavaScript 语言的形式实现新对象及其附带方法和属性的扩展。其使开发人员能够管理文档安全性，与数据库通信，处理文件附件以及操作 PDF 文件，因此其表现为交互式、web 表单等。

JavaScript action 是一种由 JavaScript 解释器编译和执行脚本的动作。类 foxit::pdf::actions::JavaScriptAction 派生自 Action，并提供接口用来获取/设置 JavaScript action 数据。

在附录中可以查看 Foxit PDF SDK 支持的 JavaScript 方法和属性列表。

Example:

3.32.1 如何添加文档级的 JavaScript 动作

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

using namespace foxit;
using namespace pdf;
using namespace annots;

// Load Document doc.
...
```

```
actions::JavaScriptAction javascript_action = (actions::JavaScriptAction)Action::Create(doc,
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"app.alert(\"Hello Foxit \");");
AdditionalAction additional_act(doc);
additional_act.SetAction(AdditionalAction::e_TriggerDocWillClose,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerDocWillClose);
...
```

3.32.2 如何添加注释级的 JavaScript 动作

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

using namespace foxit;
using namespace pdf;
using namespace annots;

// Load Document and get a widget annotation.
...

actions::JavaScriptAction javascript_action = (actions::JavaScriptAction)Action::Create(page.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"app.alert(\"Hello Foxit \");");
AdditionalAction additional_act(annot);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotMouseButtonPressed,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotMouseButtonPressed);
...
```

3.32.3 如何添加表单级的 JavaScript 动作

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace annots;
using namespace interform;

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

// Add text field.
Control control = form.AddControl(page, L"Text Field0", Field::e_TypeTextField, RectF(50, 600, 90, 640));
control.GetField().SetValue(L"3");
// Update text field's appearance.
control.GetWidget().ResetAppearanceStream();
```

```
Control control1 = form.AddControl(page, L"Text Field1", Field::e_TypeTextField, RectF(100, 600, 140, 640));
control1.GetField().SetValue(L"23");
// Update text field's appearance.
control1.GetWidget().ResetAppearanceStream();

Control control2 = form.AddControl(page, L"Text Field2", Field::e_TypeTextField, RectF(150, 600, 190, 640));
actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"AFSimple_Calculate(\"SUM\", new Array (\"Text Field0\", \"Text Field1\"));" );
Field field2 = control2.GetField();
AdditionalAction additional_act(field2);
additional_act.SetAction(AdditionalAction::e_TriggerFieldRecalculateValue,javascipt_action);
// Update text field's appearance.
control2.GetWidget().ResetAppearanceStream();
...
```

3.32.4 如何使用 JavaScript 向 PDF 页面添加一个新的注释

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

// Load Document and get form field, construct a Form object and a Filler object.
...

actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"var annot = this.addAnnot({ page : 0, type : \"Square\", rect : [ 0, 0, 100, 100 ], name :
\"UniqueID\", author : \"A. C. Robat\", contents : \"This section needs revision.\" });" );
AdditionalAction additional_act(field);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.32.5 如何使用 JavaScript 获取/设置注释的属性 (strokeColor, fillColor, readOnly, rect, type)

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

// Load Document and get form field, construct a Form object and a Filler object.
...

// Get properties of annotations.
actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"var ann = this.getAnnot(0, \" UniqueID \"); if (ann != null) { console.println(\"Found it!
type: \" + ann.type); console.println(\"readOnly: \" + ann.readOnly); console.println(\"strokeColor: \" +
ann.strokeColor);console.println(\"fillColor: \" + ann.fillColor); console.println(\"rect: \" + ann.rect);}");
AdditionalAction additional_act(field);
```

```
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);

// Set properties of annotations (only take strokeColor as an example).
actions::JavaScriptAction javascipt_action1 = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action1.SetScript(L"var ann = this.getAnnot(0, \"UniqueID\");if (ann != null) { ann.strokeColor =
color.blue; }");
AdditionalAction additional_act1(field1);
additional_act1.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action1);
additional_act1.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.32.6 如何使用 JavaScript 销毁注释

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

// Load Document and get form field, construct a Form object and a Filler object.
...

actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"var ann = this.getAnnot(0, \" UniqueID \"); if (ann != null) { ann.destroy(); } " );
AdditionalAction additional_act(field);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.33 密文 (Redaction)

密文是一种在保持文档布局的同时删除文档中敏感信息的功能。它可以帮助用户永久删除 PDF 文档中的可见文本和图片，以保护一些保密信息，如社会安全号码、信用卡信息、产品发布日期等等。

密文是一种标记注释，用于标记 PDF 文件的某些内容，标记的内容在注释被应用后会被删除。

执行密文，您可以使用如下的 APIs:

- 调用 foxit::addon::Redaction::Redaction 创建一个 redaction 模块。如果在函数 common::Library::Initialize 中使用的 license 授权信息没有定义"Redaction"，则表示用户没有权限使用 redaction 相关的函数，并且构造函数会抛出 foxit::e_ErrInvalidLicense 异常。
- 然后调用 foxit::addon::Redaction::MarkRedactAnnot 创建一个 redaction 对象，对需要进行 redaction 的页面内容 (文本对象、图片对象和路径对象) 进行标记。

- 最后调用 foxit::addon::Redaction::Apply 在标记区域应用 redaction：永久删除标记区域的文本和图形。

备注：*要使用 redaction 功能，请确保授权 key 文件中包含 'Redaction' 模块。*

Example:

3.33.1 如何将 PDF 文档第一页中的文本 "PDF" 设置为密文

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"
#include "include/addon/fs_redaction.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace addon;
using namespace pdf;
using namespace foxit::pdf::annots;
...

Redaction redaction(doc);
// Parse PDF page.
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
//Find Text Object to redact
TextPage text_page(page);
TextSearch text_search(text_page);
text_search.SetPattern(L"PDF");
RectFArray rect_array;
while(text_search.FindNext()) {
        rect_array.Append(text_search.GetMatchRects());
 }
if(rect_array.GetSize() > 0) {
        Redact redact = redaction.MarkRedactAnnot(page, rect_array);
        redact.ResetAppearanceStream();
        doc.SaveAs(output_directory + L"AboutFoxit_redected_default.pdf");

        // set border color to Green
        redact.SetBorderColor((long)0x00FF00);
        // set fill color to Blue
        redact.SetFillColor((long)0x0000FF);
        // set rollover fill color to Red
        redact.SetApplyFillColor((long)0xFF0000);
        redact.ResetAppearanceStream();
        doc.SaveAs(output_directory + L"AboutFoxit_redected_setColor.pdf");

        redact.SetOpacity((float)0.5);
        redact.ResetAppearanceStream();
        doc.SaveAs(output_directory + L"AboutFoxit_redected_setOpacity.pdf");
```

```
        if(redaction.Apply())
                cout << "Redact page(0) succeed." << endl;
        else
                cout << "Redact page(0) failed." << endl;
 }
doc.SaveAs(output_directory + L"AboutFoxit_redected_apply.pdf");
```

3.34 对比 (Comparison)

对比功能可以帮助用户查看两个版本的 PDF 文档之间的差异。Foxit PDF SDK 提供 APIs 用以逐页比较两个 PDF 文档，并返回文档间的差异。

差异可以定义为三种类型：删除、插入和替换。您可以将这些差异保存为 PDF 文件并标记为注释。

备注：*要使用对比功能，请确保授权 key 文件中包含 'Comparison' 模块。*

Example:

3.34.1 如何对比两个 PDF 文档，并将差异保存到一个 PDF 文件中

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"
#include "include/addon/fs_compare.h"
#include "include/common/fxcrt/fx_basic.h"

using namespace foxit;
using namespace common;
using namespace addon;
using namespace pdf;
using namespace foxit::pdf::annots;
...

PDFDoc base_doc("input_base_file");
ErrorCode error_code = base_doc.Load();
if (error_code != foxit::e_ErrSuccess) {
   return 1;
}

PDFDoc compared_doc("input_compared_file");
error_code = compared_doc.Load();
if (error_code != foxit::e_ErrSuccess) {
   return 1;
}

Comparison comparison(base_doc, compared_doc);

// Start comparing.
CompareResults result = comparison.DoCompare(0, 0, Comparison::e_CompareTypeText);
```

```
CompareResultInfoArray& oldInfo = result.results_base_doc;
CompareResultInfoArray& newInfo = result.results_compared_doc;
int oldInfoSize = oldInfo.GetSize();
int newInfoSize = newInfo.GetSize();
PDFPage page = compared_doc.GetPage(0);
for (int i=0; i<newInfoSize; i++)
{
   const CompareResultInfo& item = newInfo.GetAt(i);
   CompareResultInfo::CompareResultType type = item.type;
   if (type == CompareResultInfo::e_CompareResultTypeDeleteText)
   {
      String res_string;
      res_string.Format((FX_LPCSTR)"\"%s\"", (FX_LPCSTR)String::FromUnicode(item.diff_contents));
      CreateDeleteTextStamp(page, item.rect_array, 0xff0000, WString::FromLocal(res_string), L"Compare :
Delete", L"Text");
   }
   else if (type == CompareResultInfo::e_CompareResultTypeInsertText)
   {
      String res_string;
      res_string.Format((FX_LPCSTR)"\"%s\"", (FX_LPCSTR)String::FromUnicode(item.diff_contents));
      CreateDeleteText(page, item.rect_array, 0x0000ff, WString::FromLocal(res_string), L"Compare : Insert",
L"Text");
   }
   else if (type == CompareResultInfo::e_CompareResultTypeReplaceText)
   {
      String res_string;
      res_string.Format("[Old]: \"%s\"\r\n[New]: \"%s\"",
(FX_LPCSTR)String::FromUnicode(old_info.GetAt(i).diff_contents),
(FX_LPCSTR)String::FromUnicode(item.diff_contents));
      CreateSquigglyRect(page, item.rect_array, 0xe7651a, WString::FromLocal(res_string), L"Compare : Replace",
L"Text");
   }
}

// Save the comparison result to a PDF file.
compared_doc.SaveAs(output_directory + L"result.pdf");
```

***备注**：对于 CreateDeleteTextStamp，CreateDeleteText 和 CreateSquigglyRect 函数，请参考 SDK 包中 "\examples\simple_demo"文件夹下的 "**pdfcompare**" demo。*

3.35 光学字符识别 (OCR)

光学字符识别 (OCR) 能够将打印的文本图像翻译成机器可读的文本。OCR 普遍应用于将纸质文档扫描并创建成电子文档，也可用于现有的电子文档 (如 PDF 文档)。

从 9.0 版本开始，Linux x64 平台支持 OCR 功能，并且 OCR 引擎已升级，请联系 Foxit 支持团队或销售团队获取最新的引擎文件包。

本节将介绍如何使用 Foxit PDF SDK for Windows/Linux 设置 OCR 功能模块的使用环境。

3.35.1 系统需求

平台: Windows, Linux (x64)

开发语言: C, C++, Java, Python, C#

License key: license key 中包含 'OCR' 模块的权限

SDK 版本: Foxit PDF SDK for Windows (C++, Java, C#) 6.4 或更高版本; Foxit PDF SDK (C) 7.4 或更高版本; Foxit PDF SDK for Windows (Python) 8.3 或更高版本; Foxit PDF SDK for Linux x64 (C++, Java, C#, Python) 9.0 或更高版本

3.35.2 OCR 模块的试用限制

对于试用版本，您需要注意如下的三条限制要求：

1) 允许从第一次初始化 OCREngine 开始，连续试用 30 个自然日。
2) 允许从第一次初始化 OCREngine 开始，对累计不超过 5000 页的 PDF 页面可以使用 OCR。
3) PDF 页面会生成试用水印。此限制用于所有 SDK 模块。

3.35.3 OCR 资源文件

请联系 Foxit 支持团队或者销售团队以获取 OCR 资源文件包。

Windows:

下载 Windows 平台的引擎包后，将其解压到所需的目录 (比如，解压到一个名为 "**ocr_addon**"的目录)，您可以看到 OCR 的资源文件如下：

- **debugging_files:** 用于调试 OCR 工程的资源文件。这些文件不能随应用程序发布。
- **language_resource_CJK:** CJK 语言资源文件，包括 Chinese-Simplified, Chinese-Traditional, Japanese, 和 Korean。
- **language_resources_noCJK:** 除去 CJK 以外的语言资源文件，包括 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:** Win32 的库资源文件。
- **win64_lib:** Win64 的库资源文件。
- **readme.txt:** 介绍该目录下各个文件夹的作用，以及如何使用这些资源文件构建一个 OCR 的资源目录。

Linux x64:

下载 Linux 平台的引擎包后，将其解压到所需的目录 (比如，解压到一个名为 "**ocr_addon_linux**" 的目录)，您可以看到 OCR 的资源文件如下：

- **Data:** 以下语言的相关数据和资源文件：

  Chinese-Simplified, Chinese-Traditional, Japanese, Korean, 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.

- **Bin:** Linux x64 的库文件。

3.35.4 如何运行 OCR demo

Foxit PDF SDK for C++ API (Windows and Linux x64) 提供了一个 OCR demo 用来展示如何使用 Foxit PDF SDK 对 PDF 页面或者 PDF 文档进行 OCR，该 demo 位于 "\examples\simple_demo\ocr" 文件夹下。

3.35.4.1 构建一个 OCR 资源目录

在运行 OCR demo 之前，您需要首先构建一个 OCR 资源目录，然后将该目录的路径传给 **OCREngine::Initialize** 接口用来初始化 OCR 引擎。

Windows:

构建一个 Windows 平台的 OCR 资源目录，请按照如下的步骤：

1) 新建一个文件夹作为 OCR 的资源目录。比如，"D:/ocr_resources"。
2) 根据要编译的平台架构，选择相应的库资源。
   - 如果使用 **win32**，则将 "ocr_addon/win32_lib" 文件夹下的所有文件拷贝到 "D:/ocr_resources"。
   - 如果使用 **win64**，则将 "ocr_addon/win64_lib" 文件夹下的所有文件拷贝到 "D:/ocr_resources"。
3) 选择需要使用的语言资源。
   - 如果只需要使用 CIJ 语言 (Chinese-Simplified, Chinese-Traditional, Japanese, 和 Korean)，则将 "ocr_addon/language_resource_CJK" 文件夹下的所有文件拷贝到 "D:/ocr_resources"。
   - 如果只需要使用除 CIJ 以外所支持的语言，则将 "ocr_addon/language_resources_noCJK" 文件下的所有文件拷贝到 "D:/ocr_resources"。

- 如果需要使用所有支持的语言，则将 "ocr_addon/language_resource_CJK" 和 "ocr_addon/language_resources_noCJK" 文件夹下所有的文件拷贝到 "D:/ocr_resources"。

4) (可选) 如果需要调试 demo，请根据平台架构，选择相应的调试资源文件。

- 如果使用 win32，则将 "ocr_addon/debugging_files/win32" 文件夹下的文件拷贝到 "D:/ocr_resources"。
- 如果使用 win64，则将 "ocr_addon/debugging_files/win64" 文件夹下的文件拷贝到 "D:/ocr_resources"。

***备注：**"ocr_addon/debugging_files" 文件夹下的文件只用于调试，请不要随产品发布。*

Linux x64:

构建一个 Linux 平台的 OCR 资源目录，请按照如下的步骤：

1) 新建一个文件夹作为 OCR 的资源目录。比如，"/root/Desktop/ocr_resources"。

2) 将 "ocr_addon_linux" 目录下的 "**Data**" 和 "**Bin**"文件夹 拷贝到 "/root/Desktop/ocr_resources"。

   那么，OCR 资源目录的路径则设置为 "**/root/Desktop/ocr_resources/Bin**"。

备注：加载资源文件之前，需要设置环境变量，请执行：export LD_LIBRARY_PATH=/root/Desktop/ocr_resources/Bin.

3.35.4.2 配置 demo

构建 OCR 资源目录后，在 "\examples\simple_demo\ocr**ocr.cpp**" 文件中配置 demo。以下将以 Windows 平台为例，在 "**ocr.cpp**" 文件中配置 demo。对于 Linux x64 平台，其配置操作与 Windows 相似。

在 Visual Studio 中加载 Windows 平台的 OCR demo，请选择以下两种方法中的一种：

1) 在 "\examples\simple_demo" 文件夹下，根据您 Visual Studio 的版本双击 "simple_demo_vs2010.sln" 或者 "simple_demo_vs2015.sln" 或者 "simple_demo_vs2017.sln"或者 "simple_demo_vs2019.sln"或者 "simple_demo_vs2022.sln"。然后右击 ocr demo，选择 **Set as StartUp Project**。

2) 在 "\examples\simple_demo\ocr" 文件夹下，根据您 Visual Studio 的版本双击 "ocr_vs2010.vcxproj" 或者 "ocr_vs2015.vcxproj" 或者 "ocr_vs2017.vcxproj"或者 "ocr_vs2019.vcxproj"或者 "ocr_vs2022.vcxproj"。

指定 OCR 资源目录

如下所示，添加 OCR 资源目录，用以初始化 OCR 引擎。

```
77    try {
78      // "ocr_resource_path" is the path of ocr resources. Please refer to Developer Guide for more details.
79      WString ocr_resource_path = L"D:/ocr_resources"; // Add OCR resource file path here.
80
81      if (ocr_resource_path.IsEmpty()) {
82        std::cout<<"ocr_resource_path is still empty. Please set it with a valid path to OCR resource path."<<std::endl;
83        return 1;
84      }
```

设置语言

设置需要被 OCR 引擎识别的语言。使用 **OCREngine::SetLanguages** 接口设置语言，默认是英语。

```
104     // Set languages.
105     OCREngine::SetLanguages(L"English");
```

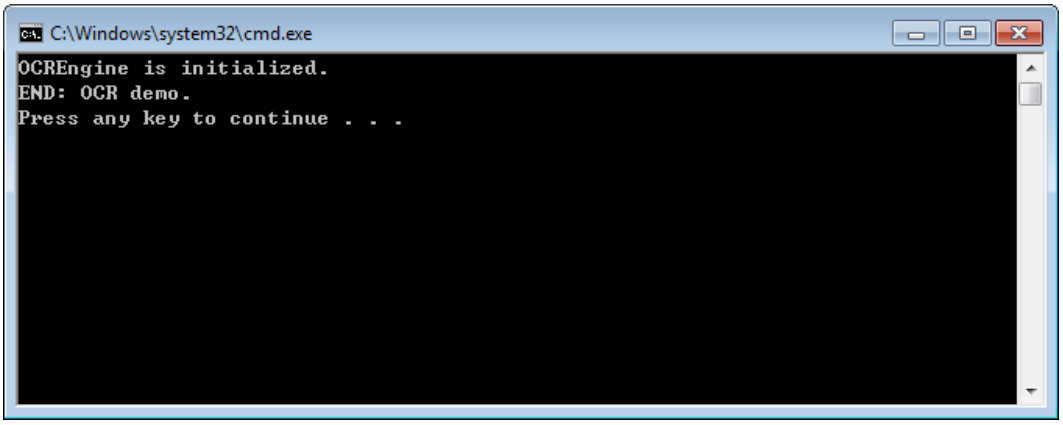
(可选) 设置 OCREngine 日志文件

如果您需要打印 OCR 引擎的整个日志记录，请取消注释 **OCREngine::SetLogFile** 函数，如下所示：

```
101     // Set log for OCREngine. (This can be opened to set log file if necessary)
102     OCREngine::SetLogFile(output_directory+L"ocr.log");
```

3.35.4.3 运行 demo

成功运行 demo 后，控制台将默认打印以下内容：



该 demo 将以四种不同的方式对默认的文档 ("\examples\simple_demo\input_files\ocr\AboutFoxit_ocr.pdf") 进行 OCR，并在输出文件夹 ("\examples\simple_demo\output_files\ocr") 下输出四个不同的 PDF 文档：

- 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

3.36 Compliance

PDF Compliance

Foxit PDF SDK 支持 PDF 版本标准化转换，当前支持的版本有 PDF 1.3, PDF 1.4, PDF 1.5, PDF 1.6 和 PDF 1.7。当转换到 PDF 1.3 版本时，如果源文档含有透明度的数据，则其会被转换到 PDF 1.4 版本而不是 PDF 1.3 版本 (PDF 1.3 版本不支持透明度)；如果源文档不含有任何透明度的数据，则会按预期转换到 PDF 1.3 版本。

PDF/A Compliance

PDF/A 是一种 ISO 标准的 PDF 文件格式版本，用于电子文档的存档和长期保存。PDF/A 与 PDF 的不同之处在于 PDF/A 禁用了 PDF 中不适合长期存档的特性，比如字体链接 (与嵌入字体相对)、加密、JavaScript、音频和视频等。

Foxit PDF SDK 提供 APIs 用以将 PDF 转换为符合 PDF/A 标准的文档，或验证 PDF 是否符合 PDF/A 标准。支持的 PDF/A 标准包括 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 和 19005-3)。

本节将介绍如何设置相关环境以运行 'compliance' demo。

3.36.1 系统需求

平台: Windows, Linux (x86 和 x64), Mac

开发语言: C, C++, Java, C#, Python, Objective-C

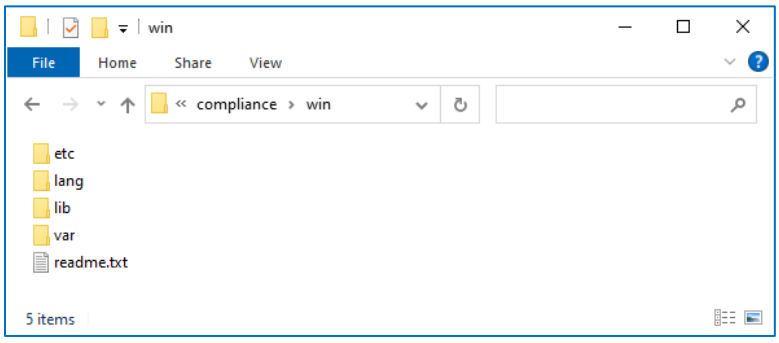
License Key: license key 中包含 'Compliance' 模块的权限
SDK 版本: Foxit PDF SDK (C++, Java, C#, Objective-C) 6.4 或更高版本 (对于 PDF Compliance, 则需要 Foxit PDF SDK 7.1 或更高版本); Foxit PDF SDK (C) 7.4 或更高版本; Foxit PDF SDK (Python) 8.3 或更高版本

3.36.2 Compliance 资源文件

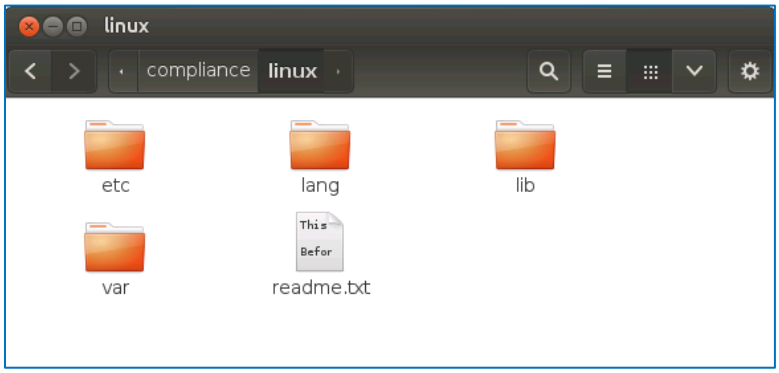
请联系 Foxit 支持团队或者销售团队以获取 Compliance 资源文件包。

获取到资源文件包后，将其解压到所需目录 (比如，Windows 解压到 "**compliance/win**"，Linux 解压到 "**compliance/linux**"，Mac 解压到 "**compliance/mac**")，然后您将看到 Compliance 的资源文件如下：

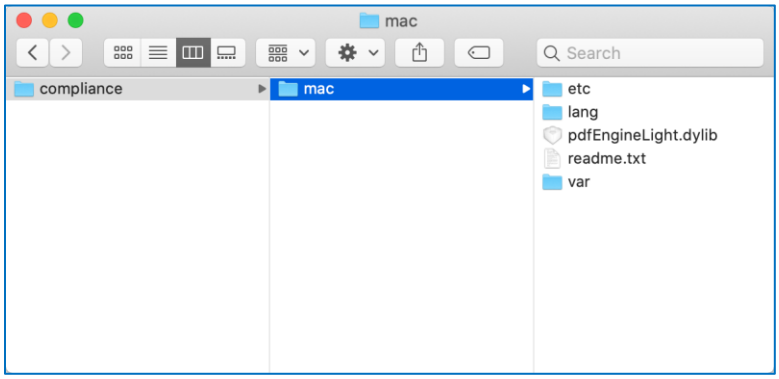
Windows 平台:



Linux 平台:



Mac 平台:



3.36.3 如何运行 compliance demo

Foxit PDF SDK 提供了一个 **compliance** demo 用来展示如何使用 Foxit PDF SDK 验证 PDF 文档是否符合 PDF/A 标准，如何将 PDF 转换为符合 PDF/A 标准的文档，以及如何进行 PDF 版本标准化转换。该 demo 位于 "\examples\simple_demo\compliance" 文件夹下。

3.36.3.1 构建一个 compliance 资源目录

在运行 **compliance** demo 之前，您需要首先构建一个 compliance 资源目录，然后将该目录的路径传给 **ComplianceEngine::Initialize** 接口用来初始化 compliance 引擎。

从 10.0 版本开始，compliance 资源文件提供了默认的线程安全机制。对于多线程，应首先调用 API **ComplianceEngine::InitializeThreadContext**，然后再使用 compliance add-on 模块中的其它接口。

Windows

在 Windows 平台构建一个 compliance 资源目录，请按照如下的步骤：

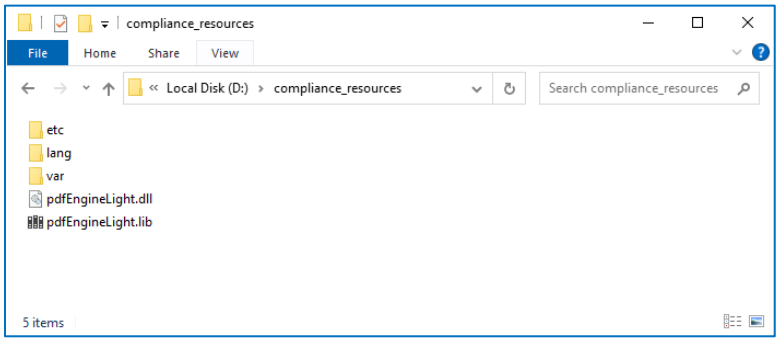
1) 新建一个文件夹作为 compliance 的资源目录。比如，"D:/compliance _resources"。

2) 将 "compliance/win" 目录下的 "**ect**"、"**lang**"、"**var**" 文件夹拷贝到 "D:/compliance_resources"。

3) 根据要编译的平台架构，选择相应的库资源。

   - 如果使用 **win32**，则将 "compliance/win/lib/x86" 文件夹下的所有文件拷贝到 "D:/compliance_resources"。
   - 如果使用 **win64**，则将 "compliance/win/lib/x64" 文件夹下的所有文件拷贝到 "D:/compliance_resources"。

例如，使用 **win32** 平台架构，则 compliance 资源目录如下所示：



Linux

在 Linux 平台构建一个 compliance 资源目录，请按照如下的步骤：

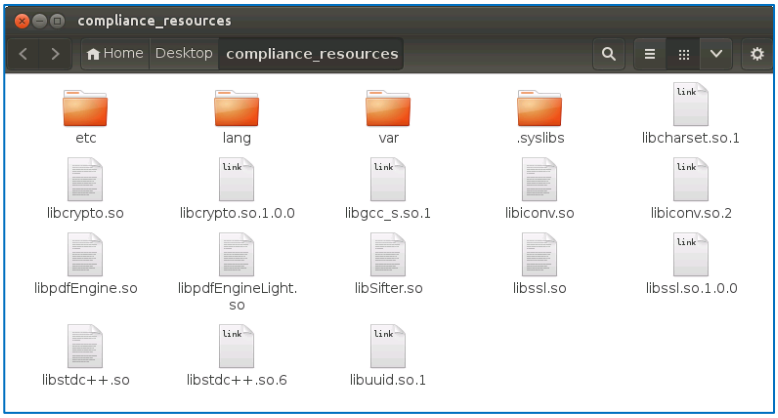
1) 新建一个文件夹作为 compliance 的资源目录。比如，"/root/Desktop/compliance_resources"。

2) 将 "compliance/linux" 目录下的 "**ect**"、"**lang**"、"**var**" 文件夹拷贝到 "/root/Desktop/compliance_resources"。

3) 根据要编译的平台架构，选择相应的库资源。

   - 如果使用 **linux32**，则将 "compliance/linux/lib/x86" 文件夹下的所有文件拷贝到 "/root/Desktop/compliance_resources"。
   - 如果使用 **linux64**，则将 "compliance/linux/lib/x64" 文件夹下的所有文件拷贝到 "/root/Desktop/compliance_resources"。

例如，使用 **linux32** 平台架构，则 compliance 资源目录如下所示：



***备注**：对于 Linux 平台，在运行 demo 之前，您需要将 compliance 资源目录加入到系统共享库目录的查找路径中，否则 **ComplianceEngine::Initialize** 会执行失败。*

例如，在运行 demo 之前，您可以通过命令 (export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/root/Desktop/compliance_resources) 将 compliance 资源目录临时加入到 LD_LIBRARY_PATH。

Mac

对于 Mac 平台，您可以直接使用 "compliance/mac" 资源文件夹作为 compliance 的资源目录。

3.36.3.2 配置 demo

构建 compliance 资源目录后，在 "\examples\simple_demo\compliance**compliance.cpp**" 文件中配置 demo。

本节以 Windows 平台为例，展示如何在 "compliance.cpp" 文件中进行 demo 配置。对于 Linux 和 Mac 平台，其配置操作与 Windows 平台相同。

在 Visual Studio 中加载 **compliance** demo (Windows)，请选择以下两种方法中的一种：

1) 在 "\examples\simple_demo" 文件夹下，根据您 Visual Studio 的版本双击 "simple_demo_vs2010.sln" 或者 "simple_demo_vs2015.sln" 或者 "simple_demo_vs2017.sln"或者 "simple_demo_vs2019.sln"或者 "simple_demo_vs2022.sln"。然后右击 **compliance** demo，选择 **Set as StartUp Project**。

2) 在 "\examples\simple_demo\compliance" 文件夹下，根据您 Visual Studio 的版本双击 "compliance _vs2010.vcxproj" 或者 "compliance_vs2015.vcxproj" 或者 "compliance_vs2017.vcxproj"或者 "compliance_vs2019.vcxproj"或者 "compliance_vs2022.vcxproj"。

指定 compliance 资源目录

在 "compliance.cpp" 文件中，如下所示，添加 compliance 资源目录，用以初始化 compliance 引擎。

```
try {
  // "compliance_resource_folder_path" is the path of compliance resource folder. Please refer to Developer Guide for more details.
  WString compliance_resource_folder_path = L"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.
  const char* compliance_engine_unlockcode = "";

  if (compliance_resource_folder_path.IsEmpty()) {
    std::cout << "compliance_resource_folder_path is still empty. Please set it with a valid path to compliance resource folder path." << std::endl;
    return 1;
  }
  // Initialize compliance engine.
  ErrorCode error_code = ComplianceEngine::Initialize(compliance_resource_folder_path, compliance_engine_unlockcode);
```

备注：

- *如果您使用的是 Foxit PDF SDK 的试用 key，则无需授权 compliance 引擎库。*
- *如果您使用的是 Foxit PDF SDK 的授权 key，则 Foxit 销售团队将向您发送一个额外的 unlock code，用于初始化 compliance 引擎库。将 unlock code 传递给初始化函数 "ComplianceEngine::Initialize (compliance_resource_folder_path, compliance_engine_unlockcode)"。*

(可选) 为 compliance engine 设置语言

ComplianceEngine::SetLanguage 函数用来为 compliance 引擎设置语言。默认的语言是英语，所有支持的语言如下所示：

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

例如，取消注释 **ComplianceEngine::SetLanguage** 函数，并将语言设置为 "Chinese-Simplified"。

```
318     // Set language. If not set language to ComplianceEngine, "English" will be used as default.
319     ComplianceEngine::SetLanguage("Chinese-Simplified");
320
```

(可选) 为 compliance 引擎设置临时文件夹

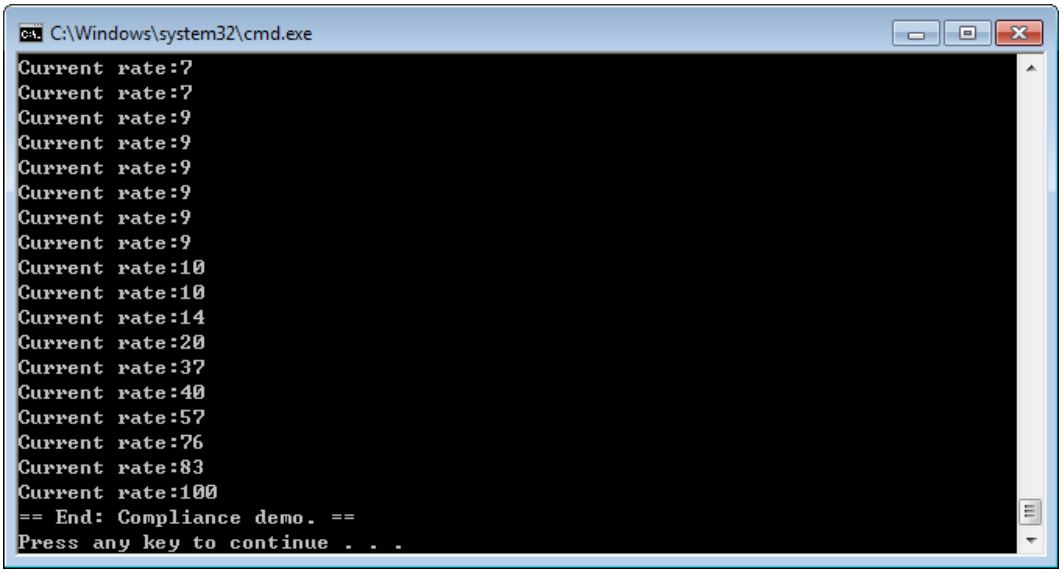
ComplianceEngine::SetTempFolderPath 函数用于设置一个临时文件夹以存储处理过程 (如验证或转换) 中生成的文件。如果此函数未设置自定义的临时文件夹，则将使用系统中默认的临时文件夹。

例如，取消注释 **ComplianceEngine::SetTempFolderPath** 函数，并将路径设置为 "D:/compliance_temp" (必须为一个有效的路径)。

```
315     // Set custom temp folder path for ComplianceEngine.
316     ComplianceEngine::SetTempFolderPath(L"D:/compliance_temp");
317
```

3.36.3.3 运行 demo

成功运行 demo 后，控制台将默认打印以下内容：



该 demo

- 验证 PDF ("\examples\simple_demo\input_files\AboutFoxit.pdf") 是否符合 PDF/A-1a 标准，并将此文档转换为符合 PDF/A-1a 标准的文档。
- 将 PDF ("\examples\simple_demo\input_files\AF_ImageXObject_FormXObject.pdf") 分别转换为 PDF-1.4 和 PDF-1.7 版本。

输出文档位于"\examples\simple_demo\output_files\compliance" 文件夹下。

3.37 优化 (Optimization)

优化功能可以通过压缩 PDF 文件中的图片、删除冗余数据，以及丢弃无用的用户数据等方式有效地减少 PDF 文件的大小，从而节省磁盘空间以及便于 PDF 文件的传输和存储。从 7.0 版本开始，优化模块提供了压缩 PDF 文件中彩色、灰度和黑白图像的方法，用于减少 PDF 文件的大小。

***备注**：要使用优化功能，请确保授权 key 文件中包含 'Optimization' 模块。*

Example:

3.37.1 如何通过压缩 PDF 文件中的彩色、灰度和黑白图像来减少 PDF 文件的大小

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/addon/optimization/fs_optimization.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon;
```

```
PDFDoc doc("input_pdf_file");
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
        printf("Error: %d\n", error_code);
        return 1;
}
Optimization_Pause pause(0,true);
addon::optimization::OptimizerSettings settings;
common::Progressive progressive =  addon::optimization::Optimizer::Optimize(doc,settings,&pause);
cout << "Optimized Start." << endl;
Progressive::State progress_state = Progressive::e_ToBeContinued;
while (Progressive::e_ToBeContinued == progress_state) {
        progress_state = progressive.Continue();
        int percent = progressive.GetRateOfProgress();
        String res_string;
        res_string.Format("Optimize progress percent: %d %",percent);
        std::cout<<res_string<<std::endl;
}
if(Progressive::e_Finished == progress_state)
{
        doc.SaveAs(L"ImageCompression_Optimized.pdf",
foxit::pdf::PDFDoc::e_SaveFlagRemoveRedundantObjects);

}
cout << "Optimized Finish." << endl;
```

3.38 HTML 转 PDF

对于一些内容比较多的 HTML 大文件或者网页，直接进行打印或者存档不太容易。Foxit PDF SDK 提供 API 接口将在线网页或者本地 HTML 文件转换为 PDF 文件，比如发票，报告等，使其更容易打印或者存档。在 HTML 转 PDF 的过程中，Foxit PDF SDK 支持在基于 HTML 的组织结构的基础上创建和添加 PDF Tag。

对于 HTML 转 PDF 模块，其支持 HTML5、CSS3 和 JavaScript。

从 7.6 版本开始，Foxit PDF SDK 支持在 Linux 平台 (只支持 x86 和 x64) 将 HTML 转化为 PDF。但是对于 Linux 平台的 HTML 转 PDF 引擎，libnss 必须是 3.22 版本的。

从 8.1 版本开始，Foxit PDF SDK 支持以文件流的形式提供 HTML2PDF 转换后生成的文件，如果您需要使用该功能，请联系 Foxit 支持团队或者销售团队以获取最新的 HTML 转 PDF 引擎文件包。

本节主要介绍如何配置运行 'html2pdf' demo 所需的环境。

3.38.1 系统需求

平台: Windows, Linux (x86 和 x64), Mac

开发语言: C, C++, Java, C#, Python, Objective-C

License Key: license key 中包含 'Conversion' 模块的权限

SDK 版本: Foxit PDF SDK (C++, Java, C#, Objective-C) 7.0 或更高版本; Foxit PDF SDK (C) 7.4 或更高版本; Foxit PDF SDK (Python) 8.3 或更高版本

3.38.2 HTML 转 PDF 引擎资源

请联系 Foxit 支持团队或者销售团队以获取 HTML 转 PDF 引擎文件包。

当获取到引擎文件包后，将其解压到所需目录 (比如，Windows 解压到 "**htmltopdf/win/**"，Linux 解压到 "**htmltopdf/linux/**"，Mac 解压到 "**htmltopdf/mac/**")。

3.38.3 如何运行 html2pdf demo

Foxit PDF SDK 提供了一个 **html2pdf** demo 用来展示如何使用 Foxit PDF SDK 将 html 文件转换为 PDF 文件。该 demo 位于 "\examples\simple_demo\html2pdf" 文件夹下。

3.38.3.1 准备一个 HTML2PDF 引擎目录

在运行 html2pdf demo 之前，您需要先将引擎包解压到一个指定的目录（比如，Windows 解压到："**D:/htmltopdf/win/**"），然后将引擎文件路径传递给 API **foxit::addon::conversion::Convert::FromHTML**，以将 html 转换为 PDF 文件。

3.38.3.2 配置 demo

对于 html2pdf demo，您可以在"\examples\simple_demo\html2pdf**html2pdf.cpp**"文件中配置 demo，或者您可以在命令行或者终端窗口中直接使用参数来对 demo 进行配置。以下将以 Windows 平台为例，在 "html2pdf.cpp" 文件中配置 demo。对于 Linux 和 Mac 平台，其配置操作与 Windows 相同。

在 Visual Studio 中加载 **html2pdf** demo (Windows)，请选择以下两种方法中的一种：

1) 在 "\examples\simple_demo" 文件夹下，根据您 Visual Studio 的版本双击 "simple_demo_vs2010.sln" 或者 "simple_demo_vs2015.sln" 或者 "simple_demo_vs2017.sln"或者 "simple_demo_vs2019.sln"或者 "simple_demo_vs2022.sln"。然后右击 **html2pdf** demo，选择 **Set as StartUp Project**。
2) 在 "\examples\simple_demo\html2pdf" 文件夹下，根据您 Visual Studio 的版本双击 "html2pdf_vs2010.vcxproj" 或者 "html2pdf_vs2015.vcxproj" 或者 "html2pdf_vs2017.vcxproj"或者 "html2pdf_vs2019.vcxproj"或者 "html2pdf_vs2022.vcxproj"。

指定 html2pdf 引擎资源目录

在 "html2pdf.cpp" 文件中，如下所示，添加 "fxhtml2pdf.exe" 引擎文件的路径，用以将 html 文件转换为 PDF 文件。

```
// "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.
WString engine_path = L"D:/htmltopdf/win/fxhtml2pdf.exe"; // or engine_path = L"D:/htmltopdf/win/fxhtml2pdf";
```

(可选) 指定 cookies 文件路径

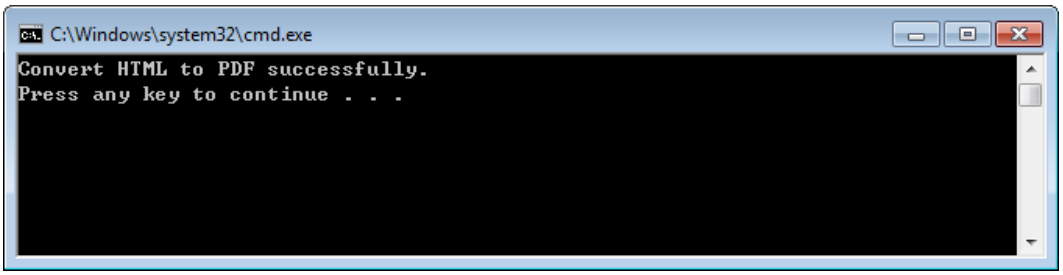
添加 cookies 文件路径，该文件是从您需要转换的 web 页面中导出的。比如，

```
// "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.
WString cookies_path = L"D:/cookies.txt";
```

3.38.3.3 运行 demo

运行 demo (不带参数)

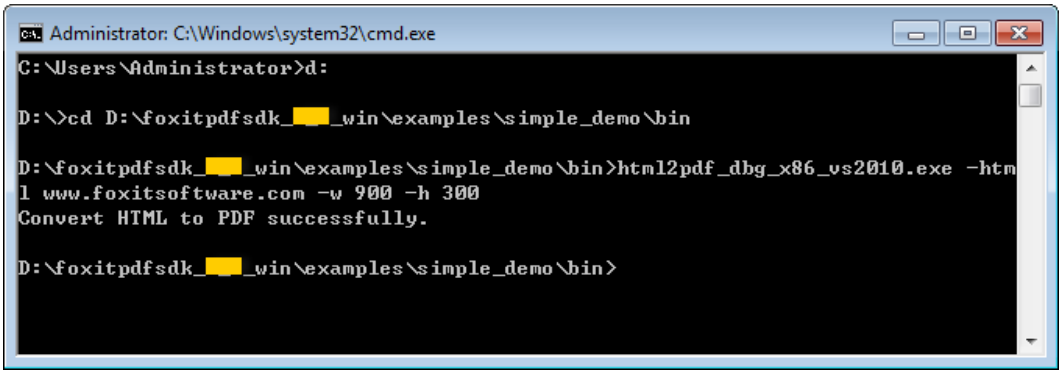
成功运行 demo 后，控制台将默认打印以下内容：



运行 demo (带参数)

成功运行 demo 后，打开一个命令行窗口，导航到 "\examples\simple_demo\bin"，输入比如 "html2pdf_dbg_x86_vs2010.exe --help" 命令查看如何使用参数来运行 demo。

例如，将 "www.foxitsoftware.com" 网页转换为 PDF 文件，并且设置转换后的 PDF 页面的宽度为 900 points，高度为 300 points。



输出文档位于"\examples\simple_demo\output_files\html2pdf" 文件夹下。

参数描述

基本语法:

html2pdf_xxx *<-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 rotation degree>] [-mode <page mode>] [-scale <scaling mode>] [-link <whether to convert link>] [-tag <whether to generate tag>] [-bookmarks <whether to generate bookmarks>] [-print_background <whether to print background>] [-optimize_tag <whether to optimize tag tree>] [-media <media style>] [-encoding <HTML encoding format>] [-render_images <Whether to render images>] [-remove_underline_for_link <Whether to remove underline for link>] [-headerfooter <Whether to generate headerfooter>] [-headerfooter_title <headerfooter title>] [-headerfooter_url <headerfooter url>] [-bookmark_root_name <bookmark root name>] [-resize_objects <Whether to enable the JavaScripts related resizing of the objects>] [-cookies <cookies file path>] [-timeout <timeout>] [--help<Parameter usage>]*

备注:

- <> 必选
- [ ] 可选

参数	**描述**
--help	参数用法的帮助信息。
-html	URL 或者 html 文件的路径。比如'-html www.foxitsoftware.com'。
-o	输出 PDF 文件的路径。
-engine	"fxhtml2pdf.exe" 引擎文件的路径。
-w	输出 PDF 文件页面的宽度，单位是 points。
-h	输出 PDF 文件页面的高度，单位是 points。
-r	输出 PDF 文件页面的旋转度。 • 0 : 0. • 1 : 90 度. • 2 : 180 度. • 3 : 270 度.
-ml	输出 PDF 文件页面的左边距。
-mr	输出 PDF 文件页面的右边距。
-mt	输出 PDF 文件页面的上边距。
-mb	输出 PDF 文件页面的下边距。
-mode	输出 PDF 文件的页面模式。 • 0 : 单页模式。 • 1 : 多页模式。
-scale	页面缩放模式。 • 0 : 不需要缩放页面。 • 1 : 将 HTML 页面缩放到输出的 PDF 的页面大小。 • 2 : 将 输出的 PDF 页面放大到 HTML 的页面大小。

参数	描述
-link	是否转换链接。 • 'yes' : 转换链接。 • 'no' : 不需要转换链接。
-tag	是否生成 tag。 • 'yes' : 生成 tag。 • 'no' : 不需要生成 tag。
-bookmarks	是否生成书签。 • 'yes' : 生成书签。 • 'no' : 不需要生成书签。
-print_background	是否打印背景。 • 'yes' : 打印背景。 • 'no' : 不需要打印背景。
-optimize_tag	是否优化 tag 树。 • 'yes' : 优化 tag 树。 • 'no' : 不需要优化 tag 树。
-media	媒体风格。 • 0 : 屏幕媒体风格。 • 1 : 印刷媒体风格。
-encoding	HTML 编码格式。 • 0 : 自动编码。 • 1-73 : 其它编码.
-render_images	是否渲染图片。 • 'yes' : 渲染图片。 • 'no' : 不需要渲染图片。
-remove_underline_for_link	是否删除链接的下划线。 • 'yes' : 删除链接的下划线。 • 'no' : 不需要删除链接的下划线。
-headerfooter	是否生成页眉页脚。 • 'yes' : 生成页眉页脚。 • 'no' : 不需要生成页眉页脚。
-headerfooter_title	页眉页脚的标题信息。
-headerfooter_url	页眉页脚的 url 信息。
-bookmark_root_name	书签根名称。
-resize_objects	是否在渲染过程中启用调整对象大小相关的 JavaScripts。 • 'yes' : 启用。

参数	描述
	• 'no' : 不启用。
-cookies	cookies 文件路径，该文件是从您需要转换的 web 页面中导出的。
-timeout	加载 web 页面的超时时间。

3.38.4 如何使用 Html2PDF API

```
#include "include/addon/conversion/fs_convert.h"

foxit::addon::conversion::HTML2PDFSettingData pdf_setting_data;
pdf_setting_data.is_convert_link = true;
pdf_setting_data.is_generate_tag = true;
pdf_setting_data.to_generate_bookmarks = true;
pdf_setting_data.rotate_degrees = foxit::common::e_Rotation0;
pdf_setting_data.page_height = 640;
pdf_setting_data.page_width = 900;
pdf_setting_data.page_mode = foxit::addon::conversion::HTML2PDFSettingData::e_PageModeSinglePage;
pdf_setting_data.scaling_mode = foxit::addon::conversion::HTML2PDFSettingData::e_ScalingModeScale;
pdf_setting_data.to_print_background = true;
pdf_setting_data.to_optimize_tag_tree = false;
pdf_setting_data.media_style = foxit::addon::conversion::HTML2PDFSettingData::e_MediaStyleScreen;
...

foxit::addon::conversion::Convert::FromHTML(url_or_html, engine_path, cookies_path, pdf_setting_data,
output_path, time_out);
```

3.38.5 如何从 stream 中获取 HTML 数据并将其转换为 PDF 文件

1. 定义一个继承自 ReaderCallback 的 FileRead 类，用于从 stream 或内存中获取 html 数据。并定义一个继承自 WriterCallback 的 FileWriter 类，用于文件写入。关于 FileRead 和 FileWriter 类的实现，请参考 "\examples\simple_demo\html2pdf" 文件夹中的 html2pdf demo。
2. 从 stream 中获取 html 数据并设置与源 html 相关的资源。
3. 调用 foxit::addon::conversion::Convert::FromHTML 函数将其转换为 PDF 文件。

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/addon/conversion/fs_convert.h"


// get HTML data from stream
if (isfilestreamload) {
  try {
    foxit::addon::conversion::HTML2PDFSettingData pdf_setting_data;
    pdf_setting_data.page_height = 650;
    pdf_setting_data.page_width = 950;
    pdf_setting_data.is_to_page_scale = false;
```

```
    pdf_setting_data.page_margin = RectF(18, 18, 18, 18);
    pdf_setting_data.is_convert_link = true;
    pdf_setting_data.rotate_degrees = foxit::common::e_Rotation0;
    pdf_setting_data.is_generate_tag = true;

    foxit::addon::conversion::HTML2PDFSettingData::HTML2PDFPageMode mode =
    foxit::addon::conversion::HTML2PDFSettingData::e_PageModeSinglePage;
    pdf_setting_data.to_generate_bookmarks = true;
    foxit::addon::conversion::HTML2PDFSettingData::HTML2PDFScalingMode scale =
    foxit::addon::conversion::HTML2PDFSettingData::e_ScalingModeNone;
    pdf_setting_data.encoding_format =
    foxit::addon::conversion::HTML2PDFSettingData::e_EncodingFormatDefault;
    pdf_setting_data.to_render_images = true;
    pdf_setting_data.to_remove_underline_for_link = false;
    pdf_setting_data.to_set_headerfooter = false;
    pdf_setting_data.headerfooter_title = L"";
    pdf_setting_data.headerfooter_url = L"";
    pdf_setting_data.bookmark_root_name = L"abcde";
    pdf_setting_data.to_resize_objects = true;
    pdf_setting_data.to_print_background = false;
    pdf_setting_data.to_optimize_tag_tree = false;
    pdf_setting_data.media_style = foxit::addon::conversion::HTML2PDFSettingData::e_MediaStyleScreen;
    pdf_setting_data.to_load_active_content = false;
    WString cookies = L"";
    WString output_path = output_directory + L"html2pdf_filestream_result.pdf";
    FileWriter* filewrite = new FileWriter();
    filewrite->LoadFile(String::FromUnicode(output_path));
    // "htmlfile" is the path of the html file to be loaded. For example: "C:/aaa.html". The method of "FromHTML"
will load this file as a stream.
    WString htmlfile = L"";
    FileReader* filereader = new FileReader();
    filereader->LoadFile(htmlfile);

    foxit::addon::conversion::HTML2PDFRelatedResourceArray html2PDFRelatedResourceArray;
    foxit::addon::conversion::HTML2PDFRelatedResource html2PDFRelatedResource;

    //"htmlfilepng" is the path of the png resource file to be loaded. For example: "C:/aaa.png". set "htmlfilepng"
in the related_resource_file of HTML2PDFRelatedResource.
    WString htmlfilepng = L"";
    FileReader* filereader1 = new FileReader();
    filereader1->LoadFile(htmlfilepng);
    html2PDFRelatedResource.related_resource_file = filereader1;
    //"relativefilepath" is  the resource file's relative path. For example: "./aaa.png".
    WString relativefilepath = L"";
    html2PDFRelatedResource.resource_file_relative_path = relativefilepath;
    html2PDFRelatedResourceArray.Add(html2PDFRelatedResource);

    foxit::addon::conversion::Convert::FromHTML(filereader, html2PDFRelatedResourceArray, engine_path,
    NULL, pdf_setting_data, filewrite, 30);
    cout << "Convert HTML to PDF successfully by filestream." << endl;
  } catch (const Exception& e) {
    cout << e.GetMessage() << endl;
```

```
  err_ret = 1;
} catch (...) {
  cout << "Unknown Exception" << endl;
  err_ret = 1;
}
```

3.39 Office 转 PDF

从 7.3 版本开始，Foxit PDF SDK 提供了 API 接口，用于在 Windows 平台上将 Microsoft Office 文档 (Word 和 Excel) 转换为专业质量 的 PDF 文档。

从 7.4 版本开始，Foxit PDF SDK 支持在 Windows 平台上将 PowerPoint 文档转换为 PDF 文档。

从 8.4 版本开始，Foxit PDF SDK 支持在 Linux 平台 (x86、x64 和 armv8) 上将 Microsoft Office 文档(Word、Excel 和 PowerPoint) 转换为专业质量的 PDF 文件。

对于使用此功能，请注意：

- 确保 Windows 系统已安装 Microsoft Office 2007 或更高版本。
- 在将 Excel 转换为 PDF 之前，请确保在 Windows 系统上已设置了默认打印机 (虚拟打印机也可以)。
- 对于 Linux x86/x64 系统，请确保已安装了 LibreOffice 。

  备注：当使用 LibreOffice 7.0 或更高版本时，如果遇到类似 "An unknown error has occurred" 的错误，您可以在运行程序之前尝试设置环境变量，如下所示：

  ```
  "export URE_BOOTSTRAP=vnd.sun.star.pathname:/opt/libreoffice7.x/program/fundamentalrc"
  ```

  其中， 'x' 代表 LibreOffice 版本号。
- 对于 Linux armv8 系统，请确保已安装了金山 WPS 办公软件。

3.39.1 系统需求

平台: Windows, Linux (x86, x64 和 armv8)

开发语言: C, C++, Python, Java, C#, Node.js

License Key: license key 中包含 'Conversion' 模块的权限
SDK 版本: Word and Excel (Foxit PDF SDK (C++, C#, Java) 7.3 或更高版本, Foxit PDF SDK (C) 7.4 或更高版本, Foxit PDF SDK (Python) 8.3 或更高版本), PowerPoint (Foxit PDF SDK (C, C++, C#, Java) 7.4 或更高版本, Foxit PDF SDK (Python) 8.3 或更高版本), Word/Excel/ PowerPoint (Foxit PDF SDK (Node.js) 10.0)

Example:

备注:

- *对于 Linux x86 和 x64，以下示例代码中的参数 "engine_path" 表示 LibreOffice 引擎的路径。要获取已安装的 LibreOffice 的路径，您可以在终端中输入 "**locate soffice.bin**" 命令来查看，比如，"/usr/lib/libreoffice/program/soffice.bin"。那么 "engine_path" 参数的值设置为 "/usr/lib/libreoffice/program"。*
- *对于 Linux armv8，以下示例代码中的参数 "engine_path" 表示 WPS 引擎 (librpcwpsapi.so) 的路径。找到 "librpcwpsapi.so" 所在的目录，比如 "/opt/kingsoft/wps-office/office6”，那么 "engine_path" 参数的值设置为 "/opt/kingsoft/wps-office/office6”。*

3.39.2 如何将 Word 文档转换为 PDF 文档

```
#include "include/addon/conversion/fs_convert.h"
...
// Make sure that SDK has already been initialized successfully.

WString word_file_path = L"test.doc";
WString output_path = L"saved.pdf";

// Use default Word2PDFSettingData values.
foxit::addon::conversion::Word2PDFSettingData word_convert_setting_data;
#if defined(_WIN32) || defined(_WIN64)
foxit::addon::conversion::Convert::FromWord(word_file_path, L"", output_path, word_convert_setting_data);
#else
foxit::addon::conversion::Convert::FromWord(word_file_path, L"", output_path, engine_path,
word_convert_setting_data);
#endif
```

3.39.3 如何将 Excel 文件转换为 PDF 文档

```
#include "include/addon/conversion/fs_convert.h"
...
// Make sure that SDK has already been initialized successfully.

WString excel_file_path = L"test.xls";
WString output_path = L"saved.pdf";

// Use default Excel2PDFSettingData values.
foxit::addon::conversion::Excel2PDFSettingData excel_convert_setting_data;
#if defined(_WIN32) || defined(_WIN64)
foxit::addon::conversion::Convert::FromExcel(excel_file_path, L"", output_path, excel_convert_setting_data);
#else
foxit::addon::conversion::Convert::FromExcel(excel_file_path, L"", output_path, engine_path,
excel_convert_setting_data);
#endif
```

3.39.4 如何将 PowerPoint 文件转换为 PDF 文档

```
#include "include/addon/conversion/fs_convert.h"
```

```
// Make sure that SDK has already been initialized successfully.

WString ppt_file_path = L"test.ppt";
WString output_path = L"saved.pdf";

// Use default PowerPoint2PDFSettingData values.
foxit::addon::conversion::PowerPoint2PDFSettingData ppt_convert_setting_data;
#if defined(_WIN32) || defined(_WIN64)
foxit::addon::conversion::Convert::FromPowerPoint(ppt_file_path, L"", output_path, ppt_convert_setting_data);
#else
foxit::addon::conversion::Convert::FromPowerPoint(ppt_file_path, L"", output_path, engine_path,
ppt_convert_setting_data);
#endif
```

3.40 输出预览 (Output Preview)

从版本 7.4 开始，Foxit PDF SDK 支持输出阅览功能，可以预览分色和测试不同的颜色配置文件。

备注：*当前，Linux ARM 平台不支持输出预览功能。*

3.40.1 系统需求

平台: Windows, Linux (x86 and x64), Mac (x64)

开发语言: C, C++, Java, C#, Python, Objective-C

License Key: 有效的 license key

SDK 版本: Foxit PDF SDK (C, C++, Java, C#, Objective-C) 7.4 或更高版本; Foxit PDF SDK (Python) 8.3 或更高版本

3.40.2 如何运行 output preview demo

在运行 "\examples\simple_demo\output_preview" 文件夹下的 output preview demo 之前，您需要首先将变量 **default_icc_folder_path** 设置为 SDK 包下 "**\res\icc_profile**" 文件夹的路径。例如：

```
// "default_icc_folder_path" is the path of the folder which contains default icc profile files. Please refer to
Developer Guide for more details.
WString default_icc_folder_path = L"E:/foxitpdfsdk_X_X_win/res/icc_profile";
```

然后，按照其他 demo 运行的步骤运行该 demo。

3.40.3 如何使用 Foxit PDF SDK 进行输出预览

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_outputpreview.h"

// Make sure that SDK has already been initialized successfully.

// Set folder path which contains default icc profile files.
Library::SetDefaultICCProfilesPath(default_icc_folder_path);
```

```
// Load a PDF document; Get a PDF page and parse it.
// Prepare a Renderer object and the matrix for rendering.

OutputPreview output_preview(pdf_doc);
WString simulation_icc_file_path = L"icc_profile.icc";
output_preview.SetSimulationProfile(simulation_icc_file_path);
output_preview.SetShowType(OutputPreview::e_ShowAll);
StringArray process_plates = output_preview.GetPlates(OutputPreview::e_ColorantTypeProcess);
StringArray spot_plates = output_preview.GetPlates(OutputPreview::e_ColorantTypeSpot);
// Set check status of process plate to be true, if there's any process plate.
for (int i = 0; i < (int)process_plates.GetSize(); i++) {
        output_preview.SetCheckStatus(process_plates[i], true);
}
// Set check status of spot plate to be true, if there's any spot plate.
for (int i = 0; i < (int)spot_plates.GetSize(); i++) {
        output_preview.SetCheckStatus(spot_plates[i], true);
}

// Generate preview bitmap
Bitmap preview_bitmap = output_preview.GeneratePreviewBitmap(pdf_page, display_matrix, renderer);
```

3.41 合并 (Combination)

合并功能用来将多个 PDF 文件合并成一个 PDF 文件。

3.41.1 如何将多个 PDF 文件合并成一个 PDF 文件

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_combination.h"

// Make sure that SDK has already been initialized successfully.

CombineDocumentInfoArray info_array;
info_array.Add(CombineDocumentInfo(input_path + L"AboutFoxit.pdf", L""));
info_array.Add(CombineDocumentInfo(input_path + L"Annot_all.pdf", L""));
info_array.Add(CombineDocumentInfo(input_path + L"SamplePDF.pdf", L""));

WString savepath = output_directory + L"Test_Combined.pdf";
uint32 option = Combination::e_CombineDocsOptionBookmark |
Combination::e_CombineDocsOptionAcroformRename |
   Combination::e_CombineDocsOptionStructrueTree | Combination::e_CombineDocsOptionOutputIntents |
   Combination::e_CombineDocsOptionOCProperties | Combination::e_CombineDocsOptionMarkInfos |
   Combination::e_CombineDocsOptionPageLabels | Combination::e_CombineDocsOptionNames |
   Combination::e_CombineDocsOptionObjectStream | Combination::e_CombineDocsOptionDuplicateStream;

Progressive progress = Combination::StartCombineDocuments(savepath, info_array, option);
Progressive::State progress_state = Progressive::e_ToBeContinued;
while (Progressive::e_ToBeContinued == progress_state) {
   progress_state = progress.Continue();
}
```

3.42 PDF Portfolio

PDF portfolios 是具有不同格式的文件的组合。Portfolio 文件本身是一个 PDF 文档，不同格式的文件可以嵌入到这种 PDF 文档中。

3.42.1 系统需求

平台: Windows, Linux, Mac

开发语言: C, C++, Java, C#, Python, Objective-C

License Key: 有效的 license key

SDK 版本: Foxit PDF SDK (C, C++, Java, C#, Objective-C) 7.6 或更高版本; Foxit PDF SDK (Python) 8.3 或更高版本

Example:

3.42.2 如何创建一个新的空白的 PDF Portfolio 文档

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_portfolio.h"
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace portfolio;

// Make sure that SDK has already been initialized successfully.

Portfolio new_portfolio = Portfolio::CreatePortfolio();

// Set properties, add file/folder node to the new portfolio.
...

// Get portfolio PDF document object.
PDFDoc portfolio_pdf_doc = new_portfolio.GetPortfolioPDFDoc();
```

3.42.3 如何从一个 PDF portfolio 文档创建一个 Portfolio 对象

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_portfolio.h"
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace portfolio;

// Make sure that SDK has already been initialized successfully.
```

```
PDFDoc portfolio_pdf_doc("portfolio.pdf");
ErrorCode error_code = portfolio_pdf_doc.Load();
if (foxit::e_ErrSuccess == error_code) {
   if (portfolio_pdf_doc.IsPortfolio()) {
      Portfolio existed_portfolio = Portfolio::CreatePortfolio(portfolio_pdf_doc);
   }
}
```

3.42.4 如何获取 portfolio nodes

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_portfolio.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_filespec.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace portfolio;

// Make sure that SDK has already been initialized successfully.

// Portfolio object has been created, assume it is named "portfolio".
...

PortfolioNode root_node = portfolio.GetRootNode();
PortfolioFolderNode root_folder(root_node);
PortfolioNodeArray sub_nodes = root_folder.GetSortedSubNodes();
for (size_t index = 0; index < sub_nodes.GetSize(); index++) {
   PortfolioNode node = sub_nodes[index];
   switch (node.GetNodeType()) {
      case PortfolioNode::e_TypeFolder: {
         PortfolioFolderNode folder_node(node);
         // Use PortfolioFolderNode's getting method to get some properties.
         ...
         PortfolioNodeArray sub_nodes_2 = folder_node.GetSortedSubNodes();
         ...
         break;
      }
      case PortfolioNode::e_TypeFile: {
         PortfolioFileNode file_node(node);
         // Get file specification object from this file node, and then get/set information from/to this file
specification object.
         FileSpec file_spec = file_node.GetFileSpec();
         ...
         break;
      }
   }
}
```

3.42.5 如何添加 file node 或者 folder node

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_portfolio.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_filespec.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace portfolio;

// Make sure that SDK has already been initialized successfully.

// Portfolio object has been created, and the root folder node has been retrieved, assume it is named
"root_folder".
...

// Add file from path.
wchar_t* path_to_a_file = L"directory/Sample.txt";
PortfolioFileNode new_file_node_1 = root_folder.AddFile(path_to_a_file);

// User can update properties of file specification for new_file_node_1 if necessary.
...

// Add file from MyStreamCallback which is inherited from StreamCallback and implemented by user.
MyStreamCallback* my_stream_callback = new MyStreamCallback();
PortfolioFileNode new_file_node_2 = root_folder.AddFile(my_stream_callback, L"file_name");

// Please get file specification of new_file_node_2 and update properties of the file specification by its setting
methods.
...

// Add a loaded PDF file.
// Open and load a PDF file, assume it is named "test_pdf_doc".
...

PortfolioFileNode new_file_node_3 = root_folder.AddPDFDoc(test_pdf_doc, L"pdf_file_name");

// User can update properties of file specification for new_file_node_3 if necessary.
...

// Add a sub folder in root_folder.
PortfolioFolderNode new_sub_foldernode = root_folder.AddSubFolder(L"Sub Folder-1");

// User can add file or folder node to new_sub_foldernode.
...
```

3.42.6 如何移除 node

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_portfolio.h"
```

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace portfolio;

// Make sure that SDK has already been initialized successfully.

// Remove a child folder node from its parent folder node.
parent_folder_node.RemoveSubNode(child_folder_node);
// Remove a child file node from its parent folder node.
parent_folder_node.RemoveSubNode(child_file_node);
```

3.43 Table Maker

从 8.4 版本开始，Foxit PDF SDK 支持向 PDF 文件中添加表格。

3.43.1 系统要求

平台: Windows, Mac, Linux
开发语言: C, C++, Java, C#, Python, Objective-C
License Key: license key 中包含 'TableMaker' 模块的权限
SDK 版本: Foxit PDF SDK 8.4 或更高版本

3.43.2 如何向 PDF 文档添加表格

在 "\examples\simple_demo\electronictable" 文件夹下，Foxit PDF SDK 提供了一个电子表格 demo，用来展示如何使用 Foxit PDF SDK 向 PDF 文档添加表格。

```
#include "include/common/fs_common.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/addon/tablegenerator/fs_tablegenerator.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon::tablegenerator;

...
// Add a spreadsheet with 4 rows and 3 columns
int index = 0;
TableCellDataArray cell_array;
for (int row = 0; row < 4; row++) {
  TableCellDataColArray col_array;
```

```
  for (int col = 0; col < 3; col++) {
    RichTextStyle style = GetTableTextStyle(index);
    WString cell_text = GetTableCellText(index++);
    TableCellData cell_data(style, cell_text, foxit::common::Image((FS_HANDLE)NULL), RectF());
    col_array.Add(cell_data);
  }
  cell_array.Add(col_array);
}
FX_FLOAT page_width = pdf_page.GetWidth();
FX_FLOAT page_height = pdf_page.GetHeight();

TableBorderInfo outside_border_left;
outside_border_left.line_width = 1;
TableBorderInfo outside_border_right;
outside_border_right.line_width = 1;
TableBorderInfo outside_border_top;
outside_border_top.line_width = 1;
TableBorderInfo outside_border_bottom;
outside_border_bottom.line_width = 1;
TableBorderInfo inside_border_row;
inside_border_row.line_width = 1;
TableBorderInfo inside_border_col;
inside_border_col.line_width = 1;
TableData data(RectF(100, 550, page_width - 100, page_height - 100), 4, 3, outside_border_left,
outside_border_right, outside_border_top, outside_border_bottom, inside_border_row, inside_border_col,
TableCellIndexArray(), FloatArray(), FloatArray());
TableGenerator::AddTableToPage(pdf_page, data, cell_array);
...
```

3.44 可访问性 (Accessibility)

从 8.4 版本开始，Foxit PDF SDK 支持对 PDF 文件进行标记 (tag)。

3.44.1 系统要求

平台: Windows, Mac, Linux
开发语言: C, C++, Java, C#, Python, Objective-C
License Key: license key 中包含 'Accessibility' 模块的权限
SDK 版本: Foxit PDF SDK 8.4 或更高版本

3.44.2 如何标记 PDF 文档

在 "\examples\simple_demo\taggedpdf" 文件夹下，Foxit PDF SDK 提供了一个标记 PDF demo，用来展示如何使用 Foxit PDF SDK 对 PDF 文档进行标记。

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/addon/accessibility/fs_taggedpdf.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace foxit::addon::accessibility;
...
PDFDoc pdfDoc(input_file);
pdfDoc.Load();
TaggedPDF taggedpdf(pdfDoc);
Progressive progressive = taggedpdf.StartTagDocument(NULL);
Progressive::State progressState = Progressive::e_ToBeContinued;
while (Progressive::e_ToBeContinued == progressState)
   progressState = progressive.Continue();
pdfDoc.SaveAs(output_file_path);
...
```

3.45 PDF 转 Office

Foxit PDF SDK 在 Windows 和 Linux 平台上提供了 API，可将 PDF 文件转换为 MS office 套件格式，同时保持原始文档的布局和格式。

3.45.1 系统要求

平台: Windows, Linux

开发语言: C, C++, Java, Python, C#

License Key: license key 中包含 'PDF2Office' 模块的权限

SDK 版本: Foxit PDF SDK for Windows (C, C++, Java, Python, C#) 9.0 或更高版本; Foxit PDF SDK for Linux (C, C++, Java, Python, C#) 9.1 或更高版本

3.45.2 PDF 转 Office 资源文件

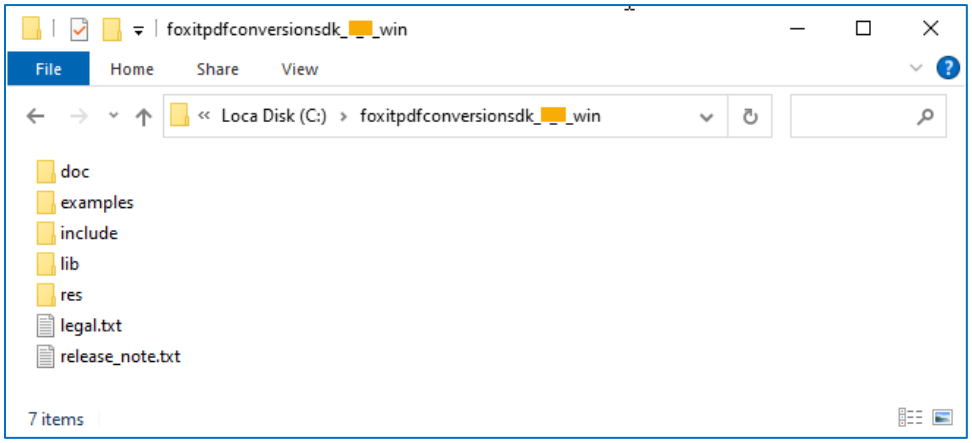
请联系 Foxit 支持团队或者销售团队以获取 PDF 转 Office 资源文件包 (Foxit PDF Conversion SDK (C++))。

备注：

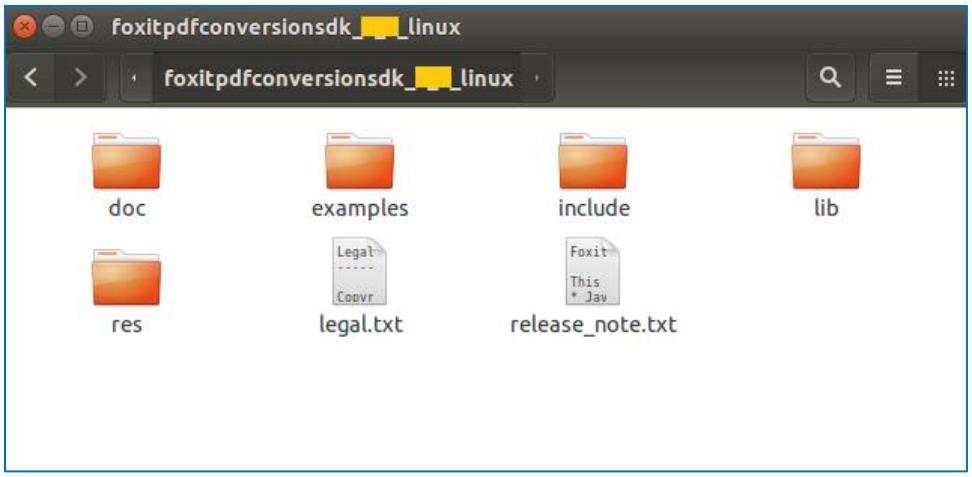
- 从版本 9.2 开始，要求使用 Foxit PDF Conversion SDK 1.5 或更高版本。
- 对于版本 9.0 / 9.1，需要使用 Foxit PDF Conversion SDK 1.4 或更低版本。

当获取到 Foxit PDF Conversion SDK 包后，将其解压到所需目录 (比如，Windows 解压到 "**/foxitpdfconversionsdk_*_win/**"，Linux x86/x64 解压到 "**/foxitpdfconversionsdk_*_Linux/**"，Linux armv7/armv8 解压到 "**/foxitpdfconversionsdk_*_Linux_arm/**")。解压后用于 PDF 转 Office 的资源文件如下所示：

Windows:



Linux x86/x64:



Linux armv7/armv8:

```
localhost:~/Desktop/foxitpdfconversionsdk_[redacted]_linux_arm # ls -l --time-style=iso
total 48
drwxr-xr-x 3 root root  4096 05-16 08:12 doc
drwxr-xr-x 3 root root  4096 05-16 08:12 examples
drwxr-xr-x 4 root root  4096 05-16 08:12 include
-rw-rw-rw- 1 root root 21005 05-16 08:12 legal.txt
drwxr-xr-x 2 root root  4096 05-16 08:12 lib
-rw-rw-rw- 1 root root  1834 05-16 08:12 release_note.txt
drwxr-xr-x 3 root root  4096 05-16 08:12 res
```

3.45.3 如何运行 pdf2office demo

Foxit PDF SDK 提供了一个 **pdf2office** demo 用来展示如何使用 Foxit PDF SDK 将 PDF 文件转换为 MS office 套件格式。该 demo 位于 "\examples\simple_demo\pdf2office" 文件夹下。

3.45.3.1 准备一个 PDF2Office 资源目录

在运行 pdf2office demo 之前，您需要先将 PDF2Office 资源文件 (Foxit PDF Conversion SDK) 解压到一个指定的目录（比如，Windows 解压到："**C:/foxitpdfconversionsdk_*_win/**"），然后将 "lib" 文件夹下的引擎文件传递给 API **PDF2Office::Initialize**，以初始化 PDF2Office 引擎。

3.45.3.2 配置 demo

对于 pdf2office demo，您可以在"\examples\simple_demo\html2pdf**pdf2office.cpp**"文件中配置 demo。以下将以 Windows 平台为例，在 "pdf2office.cpp" 文件中配置 demo。对于 Linux 平台，其配置操作与 Windows 相同。

在 Visual Studio 中加载 pdf2office demo (Windows)，请选择以下两种方法中的一种：

1) 在 "\examples\simple_demo" 文件夹下，根据您 Visual Studio 的版本双击 "simple_demo_vs2010.sln" 或者 "simple_demo_vs2015.sln" 或者 "simple_demo_vs2017.sln"或者 "simple_demo_vs2019.sln"或者 "simple_demo_vs2022.sln"。然后右击 pdf2office demo，选择 **Set as StartUp Project**。

2) 在 "\examples\simple_demo\pdf2office" 文件夹下，根据您 Visual Studio 的版本双击 "pdf2office_vs2010.vcxproj"或者 "pdf2office_vs2015.vcxproj"或者 "pdf2office_vs2017.vcxproj"或者 "pdf2office_vs2019.vcxproj"或者 "pdf2office_vs2022.vcxproj"。

指定 pdf2office 引擎目录

在 "pdf2office.cpp" 文件中，如下所示，添加 "pdf2office" 引擎文件的路径，用以将 PDF 文件转换为 office 文件。

```
// Please ensure the path is valid.
PDF2Office::Initialize(L"C:/foxitpdfconversionsdk_*_win/lib/fpdfconversionsdk_win32.dll");
```

指定 pdf2office metrics 数据文件

```
PDF2OfficeSettingData setting_data;
//Please ensure the path is valid.
setting_data.metrics_data_folder_path = L"C:/foxitpdfconversionsdk_*_win/res/metrics_data";
```

(可选) 指定是否启用基于机器学习的识别功能

```
setting_data.enable_ml_recognition = false;
```

(可选) 指定要转换的页面范围

```
setting_data.page_range = Range();
```

(可选) 指定是否转换 PDF 文档中的注释

```
setting_data.include_pdf_comments = true;
```

(可选) 对于 PDF 转 Word，指定是否保留其页面布局

```
setting_data.word_setting_data.enable_retain_page_layout = false;
```

3.45.3.3 运行 demo

成功运行 demo 后，控制台将默认打印以下内容：

```
Convert PDF file to Word format file with path.
Convert PDF file to Word format file with stream.
Convert PDF file to Excel format file with path.
Convert PDF file to Excel format file with stream.
Convert PDF file to PowerPoint format file with path.
Convert PDF file to PowerPoint format file with stream.
```

输出文档位于"\examples\simple_demo\output_files\pdf2office" 文件夹下。

3.45.4 如何使用 PDF2office API

```
#include "include/common/fs_common.h"
#include "include/addon/conversion/pdf2office/fs_pdf2office.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using namespace foxit::common::file;
using foxit::common::Library;
using namespace addon::conversion::pdf2office;

class CustomConvertCallback : public ConvertCallback {
public:
   CustomConvertCallback() {}
   ~CustomConvertCallback() {}
   virtual bool NeedToPause() {
      return true;
   }

   virtual void ProgressNotify(int converted_count, int total_count) {}
};

CustomConvertCallback callback;
Progressive progressive = PDF2Office::StartConvertToWord(input_path + L"word.pdf", NULL, output_directory +
L"pdf2word_result.docx", setting_data, &callback);
if (progressive.GetRateOfProgress() != 100) {
   Progressive::State state = Progressive::e_ToBeContinued;
   while (Progressive::e_ToBeContinued == state) {
```

```
        state = progressive.Continue();
    }
}
cout << "Convert PDF file to Word format file with path." << endl;

// Convert PDF file to Excel format file.
progressive = PDF2Office::StartConvertToExcel(input_path + L"excel.pdf", NULL, output_directory +
L"pdf2excel_result.xlsx", setting_data);
if (progressive.GetRateOfProgress() != 100) {
    Progressive::State state = Progressive::e_ToBeContinued;
    while (Progressive::e_ToBeContinued == state) {
        state = progressive.Continue();
    }
}
cout << "Convert PDF file to Excel format file with path." << endl;

// Convert PDF file to PowerPoint format file.
progressive = PDF2Office::StartConvertToPowerPoint(input_path + L"powerpoint.pdf", NULL, output_directory +
L"pdf2powerpoint_result.pptx", setting_data);
if (progressive.GetRateOfProgress() != 100) {
    Progressive::State state = Progressive::e_ToBeContinued;
    while (Progressive::e_ToBeContinued == state) {
        state = progressive.Continue();
    }
}
cout << "Convert PDF file to PowerPoint format file with path." << endl;
```

3.46 DWG 转 PDF

从 10.0 版本开始，Foxit PDF SDK 支持将 DWG 文件转换为 PDF 文件。如果您想使用这个功能，您需要联系 Foxit 的支持团队或销售团队来获取引擎文件包。

3.46.1 系统要求

平台: Windows, Linux (x86 and x64), Mac(x64)

开发语言: C, C++, Java, C#, Python, Objective-C, Node.js

License Key: license key 中包含 'DWG2PDF' 模块的权限

SDK 版本: Foxit PDF SDK 10.0

3.46.2 DWG 转 PDF 引擎文件

请联系 Foxit 支持团队或销售团队获取 DWG 转 PDF 的引擎文件包。获取包后，将其解压到所需的目录。比如，Windows 解压到 "**D:/dwgtopdf/win**"，Linux 解压到 "**dwgtopdf/linux**"，Mac 解压到 "**dwgtopdf/mac**"。

3.46.3 如何运行 dwg2pdf demo

在运行 "\examples\simple_demo\dwg2pdf" 文件夹下的 dwg2pdf demo 之前，您需要首先在 demo 代码中添加 dwg2pdf 引擎文件路径，例如：

```
// "engine_path" is the path of the engine file "dwg2pdf" which is used to convert dwg to pdf. Please refer to Developer Guide for more details.
WString engine_path = L"D:/dwgtopdf/win";
```

备注: 对于 Linux (x86 和 x64) 和 Mac x64，在运行 demo 之前，需要配置环境变量。

- 对于 Linux x86 和 x64，将 dwg2pdf 引擎文件的路径添加到 LD_LIBRARY_PATH 环境变量中。

  ```
  export LD_LIBRARY_PATH=/dwgtopdf/linux:$LD_LIBRARY_PATH
  ```

- 对于 Mac x64，将 dwg2pdf 引擎文件的路径添加到 DWG_ENGINE_PATH 环境变量中。

  ```
  export DWG_ENGINE_PATH=/dwgtopdf/mac
  ```

然后，按照其他 demo 的步骤运行该 demo。

3.46.4 如何将 DWG 文件转换为 PDF 文件

```
#include "include/addon/conversion/fs_convert.h"
...
foxit::addon::conversion::DWG2PDFSettingData pdf_setting_data;
pdf_setting_data.export_flags = foxit::addon::conversion::DWG2PDFSettingData::e_FlagEmbededTTF;
pdf_setting_data.export_hatches_type = foxit::addon::conversion::DWG2PDFSettingData::
e_DWG2PDFExportHatchesTypeBitmap;
pdf_setting_data.other_export_hatches_type = foxit::addon::conversion::DWG2PDFSettingData::
e_DWG2PDFExportHatchesTypeBitmap;
pdf_setting_data.gradient_export_hatches_type = foxit::addon::conversion::DWG2PDFSettingData::
e_DWG2PDFExportHatchesTypeBitmap;
pdf_setting_data.searchable_text_type = foxit::addon::conversion::DWG2PDFSettingData::
e_DWG2PDFSearchableTextTypeNoSearch;
pdf_setting_data.is_active_layout = false; pdf_setting_data.paper_width = 640;
pdf_setting_data.paper_heigh = 900;
...
foxit::addon::conversion::Convert::FromDWG(engine_path, dwg_file_path, output_path, pdf_setting_data);
```

3.47 OFD

OFD 文件，即开放金融文档 (Open Financial Document)，用于存储和交换数字金融文档。它们是开放的并基于 XML，专门为金融文档如合同、发票和报表设计。

OFD 文件包含结构化数据和图形元素，定义文档的布局和内容，包括文本、图片、矢量图形、注释和其它相关信息。XML 格式便于理解、处理和渲染文档内容。

OFD 文件具有多种优势，包括保证文档的完整性、安全性和互操作性。它们可以进行数字签名以确保文档的真实性，并可以加密以保护敏感信息。此外，OFD 文件还支持如表单字段和数字签名等交互式功能。

要处理 OFD 文件，需要支持 OFD 标准的 OFD viewer 或编辑器软件。这些工具允许您显示、编辑、转换和打印 OFD 文档的内容。

总的来说，OFD 文件提供了一种标准化且高效的方法来数字化呈现金融文档，简化了金融信息的交换、存储和管理。

3.47.1 系统要求

平台: Windows, Linux (x64 and armv8)

开发语言: C, C++, Java, C#, Python, Node.js

License Key: license key 中包含 'OFD' 模块的权限

SDK 版本: Foxit PDF SDK 10.0

3.47.2 OFD 引擎文件

请联系 Foxit 支持团队或销售团队获取 OFD 的引擎文件包。获取包后，将其解压到所需的目录。比如，Windows 解压到 "**D:/ofd/win**"，Linux x64 解压到 "**ofd/linux64**"，Linux arm64 解压到 "**ofd/linuxarm64**"。

3.47.3 如何运行 ofd demo

在运行 "\examples\simple_demo\ofd" 文件夹下的 ofd demo 之前，您需要首先在 demo 代码中添加 ofd 引擎文件路径，例如：

```
// Initialize OFD engine. "engine_path" is the path of ofd engine file. Please refer to Developer Guide for more details.
String engine_path = "D:/ofd/win/x64"; // For Windows x64
```

然后，按照其他 demo 的步骤运行该 demo。

3.47.4 如何实现 OFD 文件与 PDF 文件之间的转换

```
#include "include/common/fs_common.h"
#include "include/addon/conversion/fs_convert.h"

using namespace foxit::common;

// Initialize OFD engine.
Library::InitializeOFDEngine(engine_path);
WString src_ofd_path = input_path + L"wm_txttiled.ofd";
WString src_pdf_path = input_path + L"test.pdf";
// Convert PDF document to OFD document, and convert OFD document to PDF document.
```

```
{
    foxit::addon::conversion::OFDConvertParam convert_param;
    // Convert OFD document to PDF document.
    foxit::addon::conversion::Convert::FromOFD(src_ofd_path, L"", output_directory + L"ofd2pdf.pdf",
convert_param);
    // Convert PDF document to OFD document.
    foxit::addon::conversion::Convert::ToOFD(src_pdf_path, L"", output_directory + L"pdf2ofd.ofd",
convert_param);
}
Library::ReleaseOFDEngine();
```

3.47.5 如何渲染 OFD 文档页面

```
#include "include/common/fs_common.h"
#include "include/addon/ofd/fs_ofddoc.h"
#include "include/addon/ofd/fs_ofdpage.h"
#include "include/addon/ofd/fs_ofdrenderer.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace addon::ofd;

// Initialize OFD engine.
Library::InitializeOFDEngine(engine_path);

// Render OFD document to bitmap.
{
    WString render_file_path = input_path + L"content_flag.ofd";
    OFDDoc doc(render_file_path, L"");
    OFDPage ofd_page = doc.GetPage(0);
    // Get the size of the page.
    float w = ofd_page.GetWidth();
    float h = ofd_page.GetHeight();
    Bitmap bitmap = Bitmap(w, h, Bitmap::e_DIBArgb, NULL, 0);
    RectI fRect(0, h, w, 0);
    bitmap.FillRect(0xFFFFFFFF, &fRect);
    // Get the display matrix of the page.
    Matrix matrix_1 = ofd_page.GetDisplayMatrix(0, 0, w, h, CommonDefines::e_Rotation0);

    OFDRenderer ofd_render(bitmap);

    Progressive progressive = ofd_render.StartRender(ofd_page, matrix_1);
    WString sSaveFilePath = output_directory + L"renderBitmap.bmp";
    // Add the bitmap to image and save the image.
    Image image;
    image.AddFrame(bitmap);
    image.SaveAs(sSaveFilePath);
    ofd_render.Release();
    ofd_page.Release();
    doc.Release();
```

```
}
Library::ReleaseOFDEngine();
```

3.48 段落编辑 (Paragraph Editing)

Foxit PDF SDK 为开发人员提供了一套多功能工具，用于微调和自定义 PDF 文档中的文本。段落编辑模块提供了复杂的调整、合并和拆分功能，让用户能够对文档内容进行精确控制。这些功能通过直观的 UI 界面实现，便于高效编辑，确保了在管理文本段落时的无缝和定制化体验。

段落编辑功能主要围绕两个核心模块，即 **ParagraphEditing** 模块和 **JoinSplit** 模块。

ParagraphEditing 模块旨在提供各种文本编辑操作，使用户能够根据具体要求轻松执行以下操作：

- 插入文本 (Insert Text)：在特定位置插入新内容，允许定制文档的精确布局。
- 删除文本 (Delete Text)：精细删除段落或字符，实现高度定制的内容修剪。
- 修改文本 (Modify Text)：调整现有文本，包括其内容和格式，以适应不同的编辑风格。
- 格式调整 (Format Adjustment)：支持对段落格式和文本样式进行细微调整，以便更准确的排版。

JoinSplit 模块包含四种重要的操作类型，以支持更复杂的文本处理需求：

- 拼接 (Join)：整合多个文本块，增强内容布局和整体文档的一致性。
- 拆分 (Split)：精细地将文本块拆分，为管理文档的各个部分提供了灵活性。
- 链接 (Link)：在文本块之间建立连接，确保相关内容的一致性。
- 取消链接 (Unlink)：断开文本块之间的链接，提供更多的编辑控制权。

3.48.1 系统要求

平台: Windows, Linux, Mac

开发语言: C, C++, Java, C#, Python, Objective-C

License Key: license key 中包含 'AdvEdit' 模块的权限

SDK 版本: Foxit PDF SDK 10.0

3.48.2 如何使用段落编辑功能

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/addon/pageeditor/fs_paragraphediting.h"

using namespace foxit;
```

```
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon::pageeditor;

...
class FxParagraphEditingProviderCallback : public foxit::addon::pageeditor::ParagraphEditingProviderCallback {
public:

  FxParagraphEditingProviderCallback(int page_index) {
    this->current_page_index_ = page_index;
  }
  virtual ~FxParagraphEditingProviderCallback() {}

  virtual void Release() { delete this; }

  virtual foxit::Matrix GetRenderMatrix(const pdf::PDFDoc& document, int page_index) {
    PDFPage page = (PDFDoc(document)).GetPage(page_index);
    int width = static_cast<int>(page.GetWidth());
    int height = static_cast<int>(page.GetHeight());
    Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, e_Rotation0);
    return matrix;
  }
  virtual void* GetPageViewHandle(const pdf::PDFDoc& document, int page_index) {
    return NULL;
  }
  virtual foxit::RectF GetClientRect(const pdf::PDFDoc& document) {
    return foxit::RectF();
  }
  virtual float GetScale(const pdf::PDFDoc& document, int page_index) {
    return 1.0f;
  }
  virtual bool GotoPageView(const pdf::PDFDoc& document, int page_index, float left, float top) {
    return true;
  }
  virtual Int32Array GetVisiblePageIndexArray(const pdf::PDFDoc& document) {
    Int32Array page_array;
    int pageindex = this->current_page_index_;
    page_array.Add(pageindex);
    return page_array;
  }
  virtual RectF GetPageVisibleRect(const pdf::PDFDoc& document, int page_index) {
    return foxit::RectF();
  }
  virtual foxit::RectF GetPageRect(const pdf::PDFDoc& document, int page_index) {
    PDFDoc doc = document;
    PDFPage page = doc.GetPage(page_index);
    float width = page.GetWidth();
    float height = page.GetHeight();
    RectF rect;
    rect.left = 0;
    rect.bottom = height;
    rect.right = width;
```

```
        rect.top = 0;
        return rect;
    }
    virtual int GetCurrentPageIndex(const pdf::PDFDoc& document) {
        return this->current_page_index_;
    }
    virtual common::Rotation GetRotation(const pdf::PDFDoc& document, int page_index) {
        PDFPage page = (PDFDoc(document)).GetPage(page_index);
        Rotation rotation = page.GetRotation();
        return rotation;
    }
    virtual void InvalidateRect(const pdf::PDFDoc& document, int page_index, const RectFArray& invalid_rects) {

    }
    virtual void AddUndoItem(const ParagraphEditingUndoItem& undo_item) {
    }
    virtual void SetDocChangeMark(const pdf::PDFDoc& document) {
    }
    virtual void NotifyTextInputReachLimit(const pdf::PDFDoc& document, int page_index) {
    }

private:
    int current_page_index_;
};

...

PDFPage page = doc.GetPage(0);
page.StartParse();
float height = page.GetHeight();

FxParagraphEditingProviderCallback* callback = new FxParagraphEditingProviderCallback(page.GetIndex());
ParagraphEditingMgr touchup_mgr = ParagraphEditingMgr(callback, doc);

// Paragraph_editing
{
    ParagraphEditing paragraphEditing = touchup_mgr.GetParagraphEditing();
    paragraphEditing.Activate();
    paragraphEditing.StartEditing(0, PointF(95, height - 728), PointF(95, height - 728));
    paragraphEditing.SetFontSize(24);
    paragraphEditing.SetUnderline(true);
    paragraphEditing.InsertText(L"InsertText_Paragraph_editing");
    paragraphEditing.Deactivate();
    WString save_pdf_path = output_directory + L"Paragraph_editing.pdf";
    doc.SaveAs(save_pdf_path, PDFDoc::e_SaveFlagNoOriginal);
}

// Join&split
{
    JoinSplit joinSplit = touchup_mgr.GetJoinSplit();
    joinSplit.Activate();
    joinSplit.OnLButtonDown(0, PointF(289, 659));
```

```
    joinSplit.OnLButtonUp(0, PointF(289, 659));
    joinSplit.SplitBoxes();
    joinSplit.Deactivate();
    WString save_pdf_path = output_directory + L"Split_Boxes.pdf";
    doc.SaveAs(save_pdf_path, PDFDoc::e_SaveFlagNoOriginal);

    joinSplit.Activate();
    joinSplit.OnLButtonDown(0, PointF(307, height - 637));
    joinSplit.OnLButtonUp(0, PointF(307, height - 637));
    joinSplit.OnLButtonDown(0, PointF(307, height - 453));
    joinSplit.OnLButtonUp(0, PointF(307, height - 453));
    joinSplit.JoinBoxes();
    joinSplit.Deactivate();
    save_pdf_path = output_directory + L"Join_Boxes.pdf";
    doc.SaveAs(save_pdf_path, PDFDoc::e_SaveFlagNoOriginal);
}
```

3.49 3D 渲染

PDF 中的 3D 渲染可以将 3D 模型转换为 2D 图像或动画，并嵌入到文档中。3D 模型可以使用专门的 3D 建模软件或 CAD (计算机辅助设计) 软件创建，然后渲染成可以在 PDF 中查看的 2D 格式。

3D 注释是一种功能，允许用户在 PDF 中的 3D 模型的特定部分添加上下文信息。注释可以包括文本、图片，甚至是指向外部资源的链接，从而提供有关模型的更详细的信息和见解。

3.49.1 系统要求

平台: Windows

开发语言: C, C++, Java, C#, Python

License Key: license key 中包含 '3D' 模块的权限

SDK 版本: Foxit PDF SDK 10.0

***备注**：如果您正在使用 Windows 7 系统，您可能需要访问 https://support.microsoft.com/zh-cn/help/4019990/update-for-the-d3dcompiler-47-dll-component-on-windows 链接下载并安装 "D3DCOMPILER_47.dll"。否则，您可能会在使用 3D 模块时遇到错误。*

3.49.2 如何显示 3D 注释

```
#include "include/addon/3d/fs_pdf3d.h"

// Load PDF document.
...
PDF3DContext pdf_context = PDF3DContext(pdf_doc);
...

void CXXXView::On3dDisplay3dannot()
{
    // Get the 3d annotation instance array.
```

```
    PDF3DAnnotInstanceArray annot_data_arr = pdf_context.GetPage3DAnnotArray(0);
    if (annot_data_arr.GetSize() == 0) return;
    // Class parameter.
    PDF3DAnnotInstance annotData = annot_data_arr.GetAt(0);
    // Activate the canvas to display the 3d annotation. Pass in a window handle to embed canvas.
    annotData.ActivateCanvas(this->GetSafeHwnd());
}
```

3.49.3 如何设置渲染模式和 controller

```
#include "include/addon/3d/fs_pdf3d.h"

// Rotate to view 3D annotations.
annotData.SetController(PDF3DAnnotInstance::e_ControllerRotate);

// Render 3D annotations as transparent.
annotData.SetRenderMode(PDF3DAnnotInstance::e_RenderModeTransparent);
```

FAQ

1. 当在 Visual Studio 中编译 demo 时，如果遇到 "'xcopy' exited with code 9009" 错误，如何进行修复？

在 Visual Studio 编译 demo 时，如果遇到 "'xcopy' exited with code 9009" 错误，如下所示：

```
'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
```

请注意检查以下几点：

1) 检查在 *"%SystemRoot%\System32"* 目录下是否存在 **xcopy.exe**，如果没有，请从其他机器上拷贝。

2) 检查系统的 **PATH** 环境变量是否设置正确。系统 **PATH** 环境变量必须包含 *"%SystemRoot%\System32;%SystemRoot%;"*，如果 **xcopy** 的环境变量设置是正确的，但仍然报错，可以尝试将 **xcopy** 的路径放置在最前面。因为可能是某些其他的环境变量出现拼写错误，从而导致后面的环境变量都失效了。请检查确认。

检查后，在命令行中输入 xcopy 命令，如果系统能够识别，关闭 Visual Studio, 重启 demo，上述错误应该就不会出现了。

2. 如何获取 PDF 文件中指定位置的文本对象，以及更改文本对象的内容？

使用 Foxit PDF SDK 获取 PDF 文件中指定位置的文本对象以及修改文本对象的内容，请按照如下的步骤：

1) 打开一个 PDF 文件。

2) 加载 PDF 页面并获取该页面中的页面对象。

3) 使用 **PDFPage::GetGraphicsObjectAtPoint** 获取指定位置的文本对象。注意：使用页面对象获取矩形来查看文本对象的位置。

4) 更改文本对象的内容并保存 PDF 文档。

以下是示例代码:

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
```

```
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...

bool ChangeTextObjectContent()
{
    try {
            WString input_file = input_path + L"AboutFoxit.pdf";
            PDFDoc doc(input_file);
            ErrorCode error_code = doc.Load();
            if (error_code != foxit::e_ErrSuccess) {
                printf("The Doc [%s] Error: %d\n", (const char*)String::FromUnicode(input_file),
error_code);
                return false;
            }
            // Get original shading objects from the first PDF page.
            PDFPage original_page = doc.GetPage(0);
            original_page.StartParse(PDFPage::e_ParsePageNormal, NULL, false);
            foxit::PointF pointf;
            pointf.x = 92;
            pointf.y = 762;
            GraphicsObjectArray arr = original_page.GetGraphicsObjectsAtPoint (pointf, 10,
GraphicsObject::e_TypeText );
            for(int i = 0; i<arr.GetSize(); i++)
            {
                GraphicsObject* graphobj = arr.GetAt(i);
                TextObject * textobj = graphobj->GetTextObject();
                textobj->SetText(L"Foxit Test");
            }
            original_page.GenerateContent();
            WString output_directory = output_path + L"graphics_objects/";
            WString output_file = output_directory + L"After_revise.pdf";
            doc.SaveAs(output_file, PDFDoc::e_SaveFlagNormal);
    }catch (const Exception& e) {
            cout << e.GetMessage() << endl;
            return false;
    }
    return true;
}
```

3. 是否可以改变嵌入 TIFF 图像的 DPI?

无法改变。PDF 中图像的 DPI 是静态的，如果图像已经存在，Foxit PDF SDK 没有更改图像 DPI 的功能。

解决办法是您可以使用第三方库来更改图像的 DPI，然后将其添加到 PDF 中。

备注：*Foxit PDF SDK 提供了一个函数 "Image::SetDPIs"，可以用来设置图片对象的 DPI 属性，但是它仅支持使用 Foxit PDF SDK 创建或者使用 "Image::AddFrame" 函数创建的图像，不支持 JPX, GIF 和 TIF 格式。*

4. 为什么在运行 OCR 和 DWG2PDF 模块相应的 simple demo 时，即使引擎文件已经升级到最新版本并且 simple demo 已经正确配置了引擎路径，在 Windows 7 系统上仍会遇到 "无法初始化引擎文件或无法加载引擎文件" 的错误？

对于 Windows 7，您需要将引擎目录中以 **api-ms-win*** 开头的 dll 文件和 ucrtbase.dll 文件复制到系统目录中。

- 如果您使用 32 位引擎并在 32 位系统上运行，则需要将引擎目录中的 api-ms-win*.dll 文件和 ucrtbase.dll 文件复制到 "C:/Windows/System32" 中。
- 如果您使用 32 位引擎并在 64 位系统上运行，则需要将引擎目录中的 api-ms-win*.dll 文件和 ucrtbase.dll 文件复制到 "C:/Windows/SysWOW64" 中。
- 如果您使用 64 位引擎，则需要将引擎目录中的 api-ms-win*.dll 文件和 ucrtbase.dll 文件复制到 "C:/Windows/System32" 中。

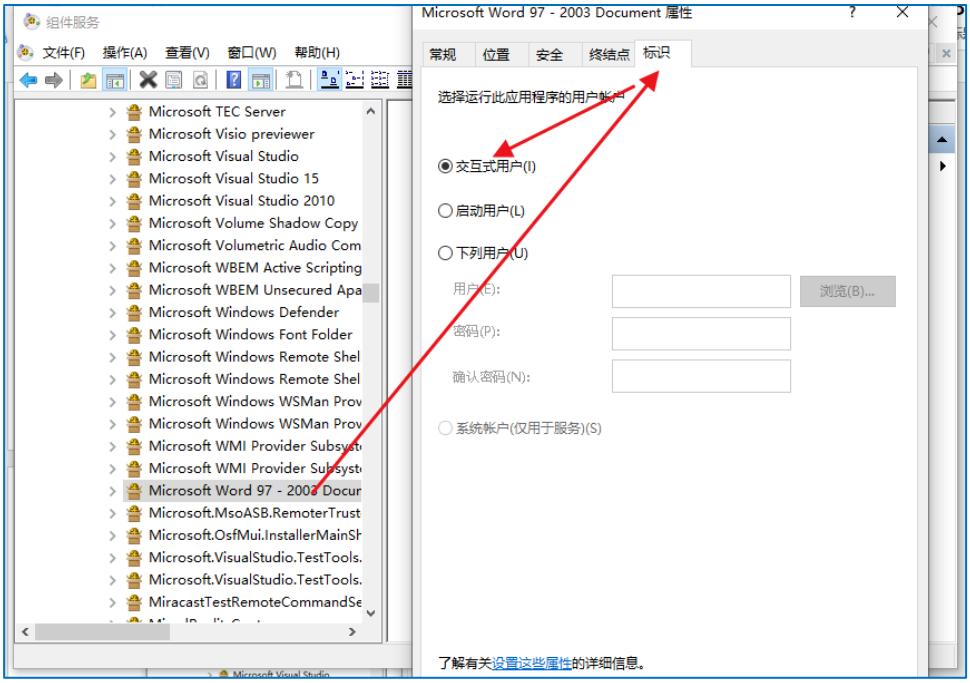
5. 如何在 Windows 服务中运行 Office2PDF 功能？

在 Windows 服务中运行 Office2PDF 功能，需要配置 office 组件服务并配置权限。

以 Word 组件为例：

1) 按下 Win+R，然后键入 **Dcomcnfg** 以打开组件服务，定位到 [组件服务] -> [计算机] -> [我的电脑] -> [DCOM 配置] -> [Microsoft Word 97-2003 文档]，右键单击并选择 "属性"。选择 [标识]，并将其设置为 "交互式用户"。

   备注：如果您使用 **Dcomcnfg** 命令无法找到 [Microsoft Word 97-2003 文档]，可以尝试使用 "**comexp.msc -32**" 命令。



2) 设置权限。点击 [安全]，将 [启动和激活权限]，[访问权限] 设置为**自定义**。点击**编辑**，添加系统当前登录账户并开启所有权限：

3) 完成以上设置后，可在 Windows 服务中运行 Word2PDF 功能。

附录

Foxit PDF SDK 支持的 JavaScript 列表

对象的属性或者方法

对象	属性/方法名称	支持的最低 SDK 版本
annotation properties	alignment	V7.0
	author	V7.0
	contents	V7.0
	creationDate	V7.0
	fillColor	V7.0
	hidden	V7.0
	modDate	V7.0
	name	V7.0
	opacity	V7.0
	page	V7.0
	readOnly	V7.0
	rect	V7.0
	richContents	V7.1
	rotate	V7.0
	strokeColor	V7.0
	textSize	V7.0
	type	V7.0
	AP	V9.0
	arrowBegin	V9.0
	arrowEnd	V9.0
	attachIcon	V9.0
	attachment	V9.0
	borderEffectIntensity	V9.0
	borderEffectStyle	V9.0
	callout	V9.0
	caretSymbol	V9.0
	dash	V9.0
	delay	V9.0
	doc	V9.0
	doCaption	V9.0
	gestures	V9.0
	inReplyTo	V9.0
	intent	V9.0
	leaderExtend	V9.0
	leaderLength	V9.0

对象	属性/方法名称	支持的最低 SDK 版本
	lineEnding	V9.0
	lock	V9.0
	noteIcon	V9.0
	noView	V9.0
	point	V9.0
	points	V9.0
	popupOpen	V9.0
	popupRect	V9.0
	print	V9.0
	quads	V9.0
	refType	V9.0
	richDefaults	V9.0
	seqNum	V9.0
	soundIcon	V9.0
	style	V9.0
	subject	V9.0
	textFont	V9.0
	toggleNoView	V9.0
	vertices	V9.0
	width	V9.0
annotation method	destroy	V7.0
	getProps	V9.0
	setProps	V9.0
	getStateInModel	V9.0
app properties	activeDocs	V4.0
	calculate	V4.0
	formsVersion	V4.0
	fs	V4.0
	fullscreen	V4.0
	language	V4.2
	platform	V4.0
	runtimeHighlight	V4.0
	viewerType	V4.0
	viewerVariation	V4.0
	viewerVersion	V4.0
	printerNames	V8.4
	runtimeHighlightColor	V8.4
	constants	V8.4
app methods	alert	V4.0
	beep	V4.0

对象	属性/方法名称	支持的最低 SDK 版本
	browseForDoc	V4.0
	clearInterval	V4.0
	clearTimeOut	V4.0
	launchURL	V4.0
	mailMsg	V4.0
	response	V4.0
	setInterval	V4.0
	setTimeOut	V4.0
	popUpMenu	V4.0
	execDialog	V8.4
	execMenuItem	V8.4
	newDoc	V8.4
	openDoc	V8.4
	popUpMenuEx	V8.4
	addMenuItem	V8.4
	addSubMenu	V8.4
	addToolButton	V8.4
	removeToolButton	V8.4
	listMenuItems	V8.4
	trustedFunction	V8.4
	beginPriv	V8.4
	endPriv	V8.4
color properties	black	V4.0
	blue	V4.0
	cyan	V4.0
	dkGray	V4.0
	gray	V4.0
	green	V4.0
	ltGray	V4.0
	magenta	V4.0
	red	V4.0
	transparent	V4.0
	white	V4.0
	yellow	V4.0
color methods	convert	V4.0
	equal	V4.0
document properties	author	V4.0
	baseURL	V4.0
	bookmarkRoot	V7.0
	calculate	V4.0

对象	属性/方法名称	支持的最低 SDK 版本
	Collab	V4.0
	creationDate	V4.0
	creator	V4.0
	delay	V4.0
	dirty	V4.0
	documentFileName	V4.0
	external	V4.0
	filesize	V4.0
	icons	V4.0
	info	V4.0
	keywords	V4.0
	modDate	V4.0
	numFields	V4.0
	numPages	V4.0
	pageNum	V4.0
	path	V4.0
	producer	V4.0
	subject	V4.0
	title	V4.0
	URL	V8.4
	dataObjects	V8.4
	hostContainer	V8.4
	templates	V8.4
	media	V8.4
	dynamicXFAForm	V8.4
	mouseX	V8.4
	mouseY	V8.4
	pageWindowRect	V8.4
	securityHandler	V8.4
	zoom	V8.4
	zoomType	V8.4
	layout	V8.4
	xfa	V8.4
document methods	addAnnot	V7.0
	addField	V4.0
	addIcon	V4.0
	calculateNow	V4.0
	deletePages	V4.0
	exportAsFDF	V4.0
	flattenPages	V7.1

对象	属性/方法名称	支持的最低 SDK 版本
	getAnnot	V7.0
	getAnnots	V7.0
	getField	V4.0
	getIcon	V4.0
	getNthFieldName	V4.0
	getOCGs	V4.0
	getPageBox	V4.0
	getPageNthWord	V4.0
	getPageNthWordQuads	V4.0
	getPageNumWords	V4.0
	getPageRotation	V7.0
	getPrintParams	V4.0
	getURL	V4.0
	importAnFDF	V4.0
	insertPages	V6.2
	mailForm	V4.0
	print	V4.0
	removeField	V4.0
	replacePages	V6.2
	resetForm	V4.0
	submitForm	V4.0
	mailDoc	V4.0
	addWatermarkFromFile	V8.4
	addWatermarkFromText	V8.4
	getPageLabel	V8.4
	setPageLabels	V8.4
	gotoNamedDest	V8.4
	saveAs	V8.4
	scroll	V8.4
	setPageTabOrder	V8.4
	selectPageNthWord	V8.4
	syncAnnotScan	V8.4
	getAnnot3D	V8.4
	getAnnots3D	V8.4
	addLink	V8.4
	removeLinks	V8.4
	getLinks	V8.4
	importIcon	V8.4
	removeIcon	V8.4
	addWeblinks	V8.4

对象	属性/方法名称	支持的最低 SDK 版本
	removeWeblinks	V8.4
	closeDoc	V8.4
	exportDataObject	V8.4
	importDataObject	V8.4
	removeDataObject	V8.4
	getDataObject	V8.4
	embedDocAsDataObject	V8.4
	createTemplate	V8.4
	removeTemplate	V8.4
	getTemplate	V8.4
	exportAsText	V8.4
	importTextData	V8.4
	exportAsXFDF	V8.4
	importAnXFDF	V8.4
	exportAsXFDFStr	V8.4
	extractPages	V8.4
	movePage	V8.4
	newPage	V8.4
	getOCGOrder	V8.4
	setOCGOrder	V8.4
	setPageBoxes	V8.4
	setPageRotations	V8.4
	setPageTransitions	V9.1
	getPageTransition	V9.1
event properties	change	V4.0
	changeEx	V4.0
	commitKey	V4.0
	fieldFull	V4.0
	keyDown	V4.0
	modifier	V4.0
	name	V4.0
	rc	V4.0
	selEnd	V4.0
	selStart	V4.0
	shift	V4.0
	source	V4.0
	target	V4.0
	targetName	V4.0
	type	V4.0
	value	V4.0

对象	属性/方法名称	支持的最低 SDK 版本
	willCommit	V4.0
event methods	add	V9.0
field properties	alignment	V4.0
	borderStyle	V4.0
	buttonAlignX	V4.0
	buttonAlignY	V4.0
	buttonFitBounds	V4.0
	buttonPosition	V4.0
	buttonScaleHow	V4.0
	buttonScaleWhen	V4.0
	calcOrderIndex	V4.0
	charLimit	V4.0
	comb	V4.0
	commitOnSelChange	V4.0
	currentValueIndices	V4.0
	defaultValue	V4.0
	doNotScroll	V4.0
	doNotSpellCheck	V4.0
	delay	V4.0
	display	V4.0
	doc	V4.0
	editable	V4.0
	exportValues	V4.0
	hidden	V4.0
	fileSelect	V4.0
	fillColor	V4.0
	lineWidth	V4.0
	highlight	V4.0
	multiline	V4.0
	multipleSelection	V4.0
	name	V4.0
	numItems	V4.0
	page	V4.0
	password	V4.0
	print	V4.0
	radiosInUnison	V4.0
	readonly	V4.0
	rect	V4.0
	required	V4.0
	richText	V4.0

对象	属性/方法名称	支持的最低 SDK 版本
	rotation	V4.0
	strokeColor	V4.0
	style	V4.0
	textColor	V4.0
	textFont	V4.0
	textSize	V4.0
	type	V4.0
	userName	V4.0
	value	V4.0
	valueAsString	V4.0
	richValue	V9.0
	submitName	V9.0
field methods	browseForFileToSubmit	V4.0
	buttonGetCaption	V4.0
	buttonGetIcon	V4.0
	buttonSetCaption	V4.0
	buttonSetIcon	V4.0
	checkThisBox	V4.0
	clearItems	V4.0
	defaultIsChecked	V4.0
	deleteItemAt	V4.0
	getArray	V4.0
	getItemAt	V4.0
	insertItemAt	V4.0
	isBoxChecked	V4.0
	isDefaultChecked	V4.0
	setAction	V4.0
	setFocus	V4.0
	setItems	V4.0
	buttonImportIcon	V9.0
	getLock	V9.0
	setLock	V9.0
	signatureGetModifications	V9.0
	signatureGetSeedValue	V9.0
	signatureInfo	V9.0
	signatureSetSeedValue	V9.0
	signatureSign	V9.0
	signatureValidate	V9.0
global methods	setPersistent	V4.0
Icon properties	name	V4.0

对象	属性/方法名称	支持的最低 SDK 版本
util methods	printd	V4.0
	printf	V4.0
	printx	V4.0
	scand	V4.0
	iconStreamFromIcon	V9.0
identity properties	loginName	V4.2
	name	V4.2
	corporation	V4.2
	email	V4.2
collab properties	user	V6.2
ocg properties	name	V6.2
ocg methods	setAction	V6.2
bookmark properties	color	V8.4
	open	V8.4
	name	V8.4
	parent	V8.4
	children	V8.4
	language	V8.4
	style	V8.4
	platform	V8.4
bookmark methods	createChild	V8.4
	insertChild	V8.4
	execute	V8.4
	setAction	V8.4
	remove	V8.4
certificate properties	binary	V8.4
	issuerDN	V8.4
	keyUsage	V8.4
	MD5Hash	V8.4
	privateKeyValidityEnd	V8.4
	privateKeyValidityStart	V8.4
	SHA1Hash	V8.4
	serialNumber	V8.4
	subjectCN	V8.4
	subjectDN	V8.4
	validityEnd	V8.4
	validityStart	V8.4
RDN properties	c	V8.4
	cn	V8.4
	e	V8.4

对象	属性/方法名称	支持的最低 SDK 版本
	l	V8.4
	o	V8.4
	ou	V8.4
	st	V8.4
security properties	handlers	V9.0
security methods	getHandler	V9.0
	importFromFile	V9.0
securityHandler properties	appearances	V9.0
	isLoggedIn	V9.0
	loginName	V9.0
	loginPath	V9.0
	name	V9.0
	uiName	V9.0
securityHandler methods	login	V9.0
	logout	V9.0
	newUser	V9.0
signatureInfo properties	objValidity	V9.0
	idValidity	V9.0
	idPrivValidity	V9.0
	docValidity	V9.0
	byteRange	V9.0
	verifyHandlerUIName	V9.0
	verifyHandlerName	V9.0
	verifyDate	V9.0
	subFilter	V9.0
	statusText	V9.0
	status	V9.0
	reason	V9.0
	name	V9.0
	mdp	V9.0
	location	V9.0
	handlerUIName	V9.0
	handlerUserName	V9.0
	handlerName	V9.0
	dateTrusted	V9.0
	date	V9.0
search properties	attachments	V9.0
	bookmarks	V9.0
	docText	V9.0
	ignoreAccents	V9.0

对象	属性/方法名称	支持的最低 SDK 版本
	markup	V9.0
	matchCase	V9.0
	matchWholeWord	V9.0
	maxDocs	V9.0
	proximity	V9.0
	stem	V9.0
	wordMatching	V9.0
	ignoreAsianCharacterWidth	V9.0
search methods	query	V9.0
	addIndex	V9.0
	removeIndex	V9.0
link properties	borderColor	V8.4
	borderWidth	V8.4
	highlightMode	V8.4
	rect	V8.4
link methods	setAction	V8.4
app.media properties	align	V8.4
	canResize	V8.4
	ifOffScreen	V8.4
	over	V8.4
	windowType	V8.4
app.media methods	createPlayer	V8.4
	openPlayer	V8.4
doc.media methods	getOpenPlayers	V8.4
Playerargs properties	doc	V8.4
	annot	V8.4
	rendition	V8.4
	URL	V8.4
	mimeType	V8.4
	settings	V8.4
	events	V8.4
MediaPlayer properties	isOpen	V8.4
	isPlaying	V8.4
	settings	V8.4
	visible	V8.4
MediaPlayer methods	close	V8.4
	play	V8.4
	seek	V8.4
	stop	V8.4
	autoPlay	V8.4

对象	属性/方法名称	支持的最低 SDK 版本
MediaSettings properties	baseURL	V8.4
	bgColor	V8.4
	bgOpacity	V8.4
	duration	V8.4
	floating	V8.4
	page	V8.4
	repeat	V8.4
	showUI	V8.4
	visible	V8.4
	volume	V8.4
	windowType	V8.4
floating properties	align	V8.4
	over	V8.4
	canResize	V8.4
	hasClose	V8.4
	hasTitle	V8.4
	title	V8.4
	ifOffScreen	V8.4
	rect	V8.4
eventListener methods	afterClose	V9.0
	afterPlay	V9.0
	afterReady	V9.0
	afterSeek	V9.0
	afterStop	V9.0
	onClose	V9.0
	onPlay	V9.0
	onReady	V9.0
	onSeek	V9.0
	onStop	V9.0
Template properties	hidden	V9.1
	name	V9.1
Template method	spawn	V9.1
span properties	alignment	V9.1
	fontFamily	V9.1
	fontStretch	V9.1
	fontWeight	V9.1
	fontStyle	V9.1
	strikethrough	V9.1
	subscript	V9.1
	superscript	V9.1

对象	属性/方法名称	支持的最低 SDK 版本
	text	V9.1
	textColor	V9.1
	textSize	V9.1
	underline	V9.1
soap properties	wireDump	V9.1
Soap method	request	V9.1
	streamDigest	V9.1
	streamEncode	V9.1
	streamFromString	V9.1
	stringFromStream	V9.1
hostContainer method	postMessage	V9.2
Fullscreen properties	transitions	V9.2
	defaultTransition	V9.2
	loop	V9.2
	timeDelay	V9.2
	useTimer	V9.2
	isFullScreen	V9.2

全局方法

方法名称	支持的最低 SDK 版本
AFNumber_Format	V4.0
AFNumber_Keystroke	V4.0
AFPercent_Format	V4.0
AFPercent_Keystroke	V4.0
AFDate_FormatEx	V4.0
AFDate_KeystrokeEx	V4.0
AFDate_Format	V4.0
AFDate_Keystroke	V4.0
AFTime_FormatEx	V4.0
AFTime_KeystrokeEx	V4.0
AFTime_Format	V4.0
AFTime_Keystroke	V4.0
AFSpecial_Format	V4.0
AFSpecial_Keystroke	V4.0
AFSpecial_KeystrokeEx	V4.0
AFSimple	V4.0
AFMakeNumber	V4.0
AFSimple_Calculate	V4.0
AFRange_Validate	V4.0
AFMergeChange	V4.0

方法名称	支持的最低 SDK 版本
AFParseDateEx	V4.0
AFExtractNums	V4.0

引用

[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 API Reference.html

备注: sdk_folder 是 SDK 包解压后的目录。

技术支持

您可以直接联系 Foxit，请使用以下的联系方式：

线上支持：

- https://www.foxitsoftware.cn/support

联系销售:

- 电话: 1-866-680-3668
- 邮箱: sales@foxitsoftware.com

联系技术支持团队：

- 电话: 1-866-MYFOXIT or 1-866-693-6948
- 邮箱: support@foxitsoftware.com