



DEVELOPER GUIDE FOXIT PDF SDK

For Node.js

TABLE OF CONTENTS

1	Introduction to Foxit PDF SDK	1
1.1	Why Choose Foxit PDF SDK.....	1
1.2	Foxit PDF SDK for Node.js	2
1.3	Evaluation	2
1.4	License	2
1.5	About this guide.....	2
2	Getting Started.....	3
2.1	System Requirements.....	3
2.2	What is in the Package	3
2.3	How to run a demo	4
2.4	How to create a simple project.....	5
3	WORKING WITH SDK API.....	7
3.1	Initialize Library	7
3.1.1	How to initialize Foxit PDF SDK.....	7
3.2	Document	7
3.2.1	How to create a PDF document from scratch.....	7
3.2.2	How to load an existing PDF document from file path	8
3.2.3	How to load an existing PDF document from a memory buffer	8
3.2.4	How to load an existing PDF document from a file read callback object.....	8
3.2.5	How to load PDF document and get the first page of the PDF document	9
3.2.6	How to save a PDF to a file.....	10
3.2.7	How to save a document into memory buffer by WriterCallback.....	10
3.3	Page.....	11
3.3.1	How to get page size.....	11
3.3.2	How to calculate bounding box of page contents.....	12
3.3.3	How to create a PDF page and set the size	12

3.3.4	How to delete a PDF page	12
3.3.5	How to flatten a PDF page.....	12
3.3.6	How to get and set page thumbnails in a PDF document.....	13
3.4	Render.....	13
3.4.1	How to render a page to a bitmap	13
3.4.2	How to render page and annotation	14
3.5	Attachment.....	14
3.5.1	How to export the embedded attachment file from a PDF and save it as a single file	14
3.5.2	How to remove all the attachments of a PDF	15
3.6	Text Page	15
3.6.1	How to extract text from a PDF page.....	16
3.6.2	How to get the text within a rectangle area in a PDF.....	16
3.7	Text Search.....	16
3.7.1	How to search a text pattern in a PDF	16
3.8	Search and Replace	17
3.8.1	System requirements	17
3.8.2	How to work with the search and replace function	17
3.9	Text Link.....	18
3.9.1	How to retrieve hyperlinks in a PDF page	18
3.10	Bookmark	18
3.10.1	How to find and list all bookmarks of a PDF.....	19
3.10.2	How to insert a new bookmark	19
3.10.3	How to create a table of contents based on bookmark information in PDFs	19
3.11	Form (AcroForm)	20
3.11.1	How to load the forms in a PDF.....	20
3.11.2	How to count form fields and get/set the properties.....	21
3.11.3	How to export the form data in a PDF to a XML file	21
3.11.4	How to import form data from a XML file	21
3.11.5	How to get coordinates of a form field.....	21
3.12	XFA Form.....	22

3.12.1	How to load XFADoc and represent an Interactive XFA form	22
3.12.2	How to export and import XFA form data.....	23
3.13	Form Filler.....	23
3.14	Form Design	23
3.14.1	How to add a text form field to a PDF	24
3.14.2	How to remove a text form field from a PDF.....	24
3.15	Annotations	24
3.15.1	General	24
3.15.2	Import annotations from or export annotations to a FDF file.....	28
3.16	Image Conversion.....	29
3.16.1	How to convert PDF pages to bitmap files	29
3.16.2	How to convert an image file to PDF file	30
3.17	Watermark.....	30
3.17.1	How to create a text watermark and insert it into the first page	31
3.17.2	How to create an image watermark and insert it into the first page.....	31
3.17.3	How to remove all watermarks from a page	32
3.18	Barcode.....	32
3.18.1	How to generate a barcode bitmap from a string	32
3.19	Security	33
3.19.1	How to encrypt a PDF file with Foxit DRM.....	33
3.20	Reflow	34
3.20.1	How to create a reflow page and render it to a bmp file	34
3.21	Asynchronous PDF	35
3.22	Pressure Sensitive Ink.....	35
3.22.1	How to create a PSI and set the related properties for it.....	35
3.23	Wrapper	36
3.23.1	How to open a document including wrapper data	36
3.24	PDF Objects	36
3.24.1	How to remove some properties from catalog dictionary.....	36

3.25	Page Object	37
3.25.1	How to create a text object in a PDF page	37
3.25.2	How to add an image logo to a PDF page	38
3.26	Marked content	38
3.26.1	How to get marked content in a page and get the tag name	38
3.27	Layer	39
3.27.1	How to create a PDF layer	39
3.27.2	How to set all the layer nodes information.....	39
3.27.3	How to edit layer tree	40
3.28	Signature.....	40
3.28.1	How to sign the PDF document with a signature	41
3.29	PAdES	42
3.30	PDF Action	42
3.30.1	How to create a URI action and insert to a link annot	42
3.30.2	How to create a GoTo action and insert to a link annot.....	43
3.31	JavaScript	43
3.31.1	How to add JavaScript Action to Document	44
3.31.2	How to add JavaScript Action to Annotation.....	44
3.31.3	How to add JavaScript Action to FormField.....	44
3.31.4	How to add a new annotation to PDF using JavaScript	45
3.31.5	How to get/set properties of annotations (strokeColor, fillColor, readOnly, rect, type) using JavaScript	45
3.31.6	How to destroy annotation using JavaScript.....	46
3.32	Redaction	46
3.32.1	How to redact the text "PDF" on the first page of a PDF.....	47
3.33	Comparison.....	48
3.33.1	How to compare two PDF documents and save the differences between them into a PDF file	48
3.34	OCR.....	49
3.34.1	System requirements	49

3.34.2	Trial limit for SDK OCR add-on module	50
3.34.3	OCR resource files.....	50
3.34.4	How to run the OCR demo.....	51
3.35	Compliance.....	53
3.35.1	System requirements	55
3.35.2	Compliance resource files.....	55
3.35.3	How to run the compliance or preflight demo.....	56
3.36	Optimization.....	59
3.36.1	How to optimize PDF files by compressing the color/grayscale/monochrome images	60
3.37	HTML to PDF Conversion.....	60
3.37.1	System requirements	61
3.37.2	HTML to PDF engine files	61
3.37.3	How to run the html2pdf demo	61
3.37.4	How to work with Html2PDF API	65
3.37.5	How to get HTML data from stream and convert it to a PDF file	65
3.38	Office to PDF Conversion with third-party engines	67
3.38.1	System requirements	67
3.38.2	How to convert Word to PDF.....	68
3.38.3	How to convert Excel to PDF.....	68
3.38.4	How to convert PowerPoint to PDF.....	68
3.39	Office to PDF Conversion without third-party engines	69
3.39.1	System requirements	69
3.39.2	Office to PDF resource files (Foxit PDF Conversion SDK)	69
3.39.3	How to run the office2pdf demo using Foxit PDF Conversion SDK	69
3.39.4	How to convert office files to PDF without third-party engines	70
3.40	Output Preview.....	70
3.40.1	System requirements	70
3.40.2	How to run the output preview demo.....	71
3.40.3	How to do output preview using Foxit PDF SDK.....	71
3.41	Combination.....	71

3.41.1	How to combine several PDF files into one PDF file	72
3.42	PDF Portfolio	72
3.42.1	System requirements	72
3.42.2	How to create a new and blank PDF portfolio	73
3.42.3	How to create a Portfolio object from a PDF portfolio	73
3.42.4	How to get portfolio nodes	73
3.42.5	How to add file node or folder node	74
3.42.6	How to remove a node	75
3.43	Table Maker.....	75
3.43.1	System requirements	75
3.43.2	How to add table to a PDF document.....	75
3.44	Accessibility	76
3.44.1	System requirements	76
3.44.2	How to tag a PDF document	76
3.45	PDF to Office Conversion	77
3.45.1	System requirements	77
3.45.2	PDF to Office resource files.....	77
3.45.3	How to run the pdf2office demo	78
3.45.4	How to work with PDF2office API	79
3.46	DWG to PDF Conversion	80
3.46.1	System requirements	80
3.46.2	DWG To PDF engine files.....	80
3.46.3	How to run the dwg2pdf demo	81
3.46.4	How to convert DWG to PDF.....	81
3.47	OFD.....	81
3.47.1	System requirements	82
3.47.2	OFD engine file	82
3.47.3	How to run the ofd demo.....	82
3.47.4	How to implement the conversion between OFD file and PDF file.....	82
3.47.5	How to render OFD page	83

FAQ	84
Appendix	87
Supported JavaScript List.....	87
References	101
Support.....	102

1 INTRODUCTION TO FOXIT PDF SDK

Have you ever thought about building your own application that can do everything you want with PDF files? If your answer is "Yes", congratulations! You just found the best solution in the industry that allows you to build stable, secure, efficient and full-featured PDF applications.

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

1.1 Why Choose Foxit PDF SDK

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

- **Easy to integrate**

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

- **Lightweight footprint**

Does not exhaust system resource and deploys quickly.

- **Cross-platform support**

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

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

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

- **Premium World-side Support**

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

1.2 Foxit PDF SDK for Node.js

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

In this guide, we focus on the introduction of Foxit PDF SDK for Node.js on Windows and Linux platforms.

Foxit PDF SDK for Node.js ships with simple-to-use APIs that can help Node.js developers seamlessly integrate powerful PDF technology into their own projects on Windows and Linux platforms. It provides rich features on PDF documents, such as PDF viewing, bookmark navigating, text selecting/copying/searching, PDF signatures, PDF forms, rights management, PDF annotations, and full text search.

1.3 Evaluation

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

1.4 License

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

1.5 About this guide

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

2 GETTING STARTED

It is very easy to setup Foxit PDF SDK and see it in action! This guide will provide you with a brief introduction about our SDK package. The following sections introduce the contents of system requirements, the installation package as well as how to run a demo, and create your own project.

2.1 System Requirements

Platform	System Requirement	Note
Windows	Windows Vista, 7, 8 and 10 (32-bit and 64-bit) Windows Server 2003, 2008 and 2012 (32-bit and 64-bit)	It only supports for Windows 8/10 classic style, but not for Store App or Universal App. Prerequisites for Node.js 8-20 version.
Linux	64-bit OS <ul style="list-style-type: none">The minimum supported version of the GCC compiler is gcc4.9.4.The minimum supported version of GLIBC is GLIBC_2.17.	Prerequisites for Node.js 8-20 version.

2.2 What is in the Package

Package for Windows Node.js is named "foxitpdfsdk_10_1_win_nodejs_example.zip", and package for Linux (x64) Node.js is named "foxitpdfsdk_10_1_linux64_nodejs_example.zip". They have the similar structure, so in this guide mainly introduce "foxitpdfsdk_10_1_win_nodejs_example.zip" as an example.

Download the package for Windows Node.js and extract it to a new directory like "foxitpdfsdk_10_1_win_nodejs_example.zip", which is shown in Figure 2-1.

NOTE: the highlighted rectangle in the figure is just the version of Foxit PDF SDK. Here the SDK version is 10.1, so it shows 10_1 in the package name. Other highlighted rectangles in other figures have the same meaning in this guide.

This package contains the following folders:

- doc:** API references, developer guide
- examples:** sample projects and demos
- res:** the default icc profile files used for output preview demo

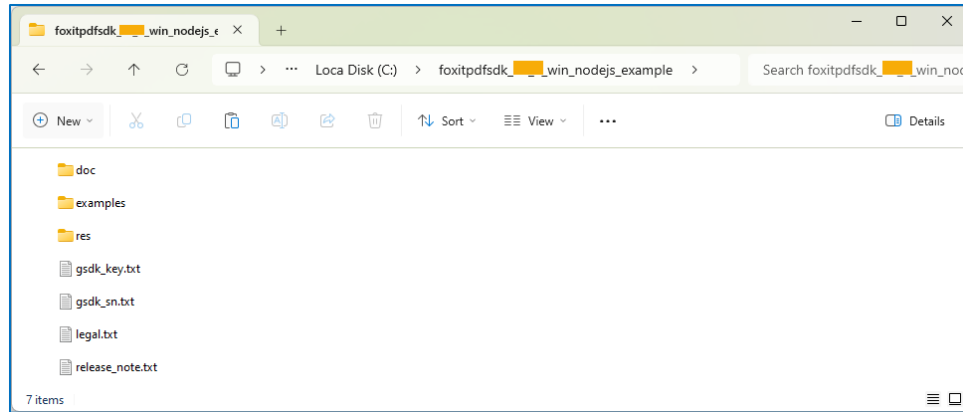


Figure 2-1

2.3 How to run a demo

Foxit PDF SDK provides several simple demos in directory "\examples\simple_demo". All these demos (except html2pdf, ocr, compliance, preflight, office2pdf, pdf2office, output preview, dwg2pdf, and ofd demos) can be run directly following the steps below:

- a. Open a command prompt or a terminal window, navigate to the root directory "foxitpdfsdk_10_1_win_nodejs_example" for Windows, or "foxitpdfsdk_10_1_linux64_nodejs_example" for Linux x64, install Foxit PDF SDK nodejs module using the command below:

```
npm i @foxitsoftware/foxit-pdf-sdk-node
```

- b. Go to directory "\examples\simple_demo", and run all demos by "RunAllDemo.bat" for Windows or by "RunAllDemo.sh" for Linux x64.
- c. If you want to run a specific single demo, please locate to the directory of the demo, for example locate to "\examples\simple_demo\annotation", and run this demo by "node annotation.js".

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

OCR and Compliance/Preflight demos

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

HTML to PDF demo

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

Office to PDF demo

For **office2pdf** demo, you need to refer to section 3.38 "[Office to PDF Conversion](#)".

Output Preview demo

For **output preview** demo, you should set the folder path which contains default icc profile files. For more details about how to run the demo, please refer to section 3.39 "[Output Preview](#)".

PDF to Office demo

For **pdf2office** demo, you should contact Foxit support team or sales team to get the engine files package for converting from PDF to office at first. For more details about how to run the demo, please refer to section 3.44 "[PDF to Office Conversion](#)".

Dwg to PDF demo

For **dwg2pdf** demo, you should contact Foxit support team or sales team to get the engine files package for converting from DWG to PDF at first. For more details about how to run the demo, please refer to section 3.45 "[DWG to PDF Conversion](#)".

OFD demo

For **ofd** demo, you should contact Foxit support team or sales team to get the OFD engine files package at first. For more details about how to run the demo, please refer to section 3.46 "[OFD](#)".

2.4 How to create a simple project

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

- 1) Create a new project folder named "test".
- 2) Copy the "SamplePDF.pdf" from the "/example/simple_demo/input_files" to the folder "test".
- 3) Run the command "npm i @foxitsoftware/foxit-pdf-sdk-node" to install the Foxit PDF SDK nodejs module.

- 4) Add the following Nodejs script file "test.js" to the folder "test".

Note:

- Set the value of "sn" in test.js with the string after "SN=" from "gsdk_sn.txt".
- Set the value of "key" in test.js with the string after "Sign=" from "gsdk_key.txt".

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

# Assuming PDFDoc doc has been loaded.

# 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=").
var sn = " "
var key = " "
FSDK.Library.Initialize(sn, key);

// Load a PDF document, and parse the first page of the document.
let doc = new FSDK.PDFDoc("SamplePDF.pdf");
let error_code = doc.Load("");
if(error_code!= FSDK.e_ErrSuccess) {
    return;
}

let page = doc.GetPage(0);
page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false);

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

// Prepare a bitmap for rendering.
let bitmap = new FSDK.Bitmap(width, height, Bitmap.e_DIBArgb, null, 0);
bitmap.FillRect(0xFFFFFFFF, null);
// Render page.
render = Renderer(bitmap, false)
render.StartRender(page, matrix, null)

// Add the bitmap to image and save the image.
let img = new FSDK.Image();
img.AddFrame(bitmap);
img.SaveAs("testpage.jpg");
FSDK.Library.Release();
```

- 5) Run "test.js" on the CMD as "node test.js", and then the "testpage.jpg" will be generated in current folder.

3 WORKING WITH SDK API

In this section, we will introduce a set of major features and list some examples for each feature to show you how to integrate powerful PDF capabilities with your applications using Foxit PDF SDK Node.js API.

3.1 Initialize Library

It is necessary for applications to initialize Foxit PDF SDK before calling any APIs. The function `FSDK.Library.Initialize` is provided to initialize Foxit PDF SDK. A license should be purchased for the application and pass unlock key and code to get proper supports. When there is no need to use Foxit PDF SDK any more, please call function `FSDK.Library.Release` to release it.

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

Example:

3.1.1 How to initialize Foxit PDF SDK

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

let error_code = FSDK.Library.Initialize(sn, key);
if (FSDK.e_ErrSuccess !== error_code) {
    return 1;
}
```

3.2 Document

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

Example:

3.2.1 How to create a PDF document from scratch

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let doc = new FSDK.PDFDoc("Sample.pdf");
```

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

3.2.2 How to load an existing PDF document from file path

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...

let doc = new FSDK.PDFDoc("Sample.pdf");
let error_code = doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    return 1;
}
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
const fs = require('fs');
const os = require('os');
const path = require('path');
const util = require('util');

...

const data = fs.readFileSync("blank.pdf");
const file_size = data.length;
const buffer = Buffer.alloc(file_size);
data.copy(buffer);
let doc = new FSDK.PDFDoc(buffer, file_size);
error_code = doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    return;
}
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
const fs = require('fs');
const os = require('os');
const path = require('path');
const util = require('util');

...

class FileReader {
    constructor() {
        this.file_ = null;
        this.file_path_ = "";
        this.hint_data_record_ = [];
    }

    LoadFile(file_path) {
        try {
            this.file_path_ = file_path;
            this.file_ = fs.openSync(file_path, 'r');
            return true;
        } catch (err) {
            console.error(err);
        }
    }
}
```



```
        return false;
    }
}

CloseFile() {
    fs.closeSync(this.file_);
}

GetSize() {
    try {
        let stats = fs.statSync(this.file_path_);
        return stats.size;
    } catch (err) {
        console.error(err);
        return 0;
    }
}

ReadBlock(offset, size_t) {
    try {
        let buffer = Buffer.alloc(size_t);
        fs.readSync(this.file_, buffer, 0, size_t, offset);
        return [true, buffer]
    } catch (err) {
        return [false, null];
    }
}

Release() { }
}
...
let input_pdf_path = "Sample.pdf";
let file_reader = new FileReader(offset);
file_reader.LoadFile(input_pdf_path);
let reader_callback = new FSDK.FileReaderCallback(file_reader);

let doc_real = new FSDK.PDFDoc(reader_callback, false);
code = doc.Load("");
if (code != FSDK.e_ErrSuccess) {
    return;
}
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let doc = new FSDK.PDFDoc("Sample.pdf");
let error_code = doc.Load("");
if (error_code != FSDK.e_ErrSuccess) {
    return 1;
}
```

```
let page = doc.GetPage(0);
page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false);
```

3.2.6 How to save a PDF to a file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let doc = new FSDK.PDFDoc("Sample.pdf");
let error_code = doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    return 1;
}
doc.SaveAs("new_Sample.pdf", FSDK.PDFDoc.e_SaveFlagNoOriginal);
```

3.2.7 How to save a document into memory buffer by WriterCallback

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
const fs = require('fs');
const os = require('os');
const path = require('path');
const util = require('util');

...
class FileWriter {
    constructor() {
        this.m_fileFP = null;
        this.file_path_ = "";
    }

    LoadFile(file_path) {
        try {
            this.file_path_ = file_path;
            this.m_fileFP = fs.openSync(file_path, 'w');
            if (!this.m_fileFP) {
                return false;
            }
            return true;
        } catch (err) {
            console.error(err);
            return false;
        }
    }

    GetSize() {
        try {
            let stats = fs.statSync(this.file_path_);
            return stats.size;
        } catch (err) {
            console.error(err);
            return 0;
        }
    }
}
```

```

}

Flush() {
  return true;
}

WriteBlock(buffer, offset, size) {
  try {
    const write_size = fs.writeSync(this.m_fileFP, buffer, 0, size, offset);
    if (write_size == size) {
      return true;
    } else {
      return false;
    }
  } catch (err) {
    console.error(err);
    return false;
  }
}

Release() {
  try {
    fs.closeSync(this.m_fileFP);
    this.m_fileFP = null;
  } catch (err) {
    console.error(err);
  }
}
}
}
let filewrite = new FileWriter();
let write_callback = new FSDK.FileWriterCallback(filewrite);
doc.SaveAs(write_callback, FSDK.PDFDoc.e_SaveFlagNoOriginal);
...

```

3.3 Page

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

Example:

3.3.1 How to get page size

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```

...

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

3.3.2 How to calculate bounding box of page contents

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
  
...  
// Assuming PDFDoc doc has been loaded.  
// Assuming PDFPage page has been loaded and parsed.  
let ret = page.CalcContentBBox(FSDK.PDFPage.e_CalcContentsBox);  
...
```

3.3.3 How to create a PDF page and set the size

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
  
...  
// Assuming PDFDoc doc has been loaded.  
let page = doc.InsertPage(index, PageWidth, PageHeight)
```

3.3.4 How to delete a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
...  
// Assuming PDFDoc doc has been loaded.  
  
// Remove a PDF page by page index.  
doc.RemovePage(index)  
  
// Remove a specified PDF page.  
doc.RemovePage(page)  
...
```

3.3.5 How to flatten a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
...  
  
let page = new FSDK.PDFPage();  
// Assuming PDFPage page has been loaded and parsed.  
// Flatten all contents of a PDF page.  
page.Flatten(true, FSDK.PDFPage.e_FlattenAll)  
  
// Flatten a PDF page without annotations.  
page.Flatten(true, FSDK.PDFPage.e_FlattenNoAnnot)  
  
// Flatten a PDF page without form controls.  
page.Flatten(true, FSDK.PDFPage.e_FlattenNoFormControl)  
  
// Flatten a PDF page without annotations and form controls (Equals to nothing to be flattened).  
page.Flatten(true, FSDK.PDFPage.e_FlattenNoAnnot | PDFPage.e_FlattenNoFormControl)
```

...

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFPage page has been loaded and parsed.

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

3.4 Render

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

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

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

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

Example:

3.4.1 How to render a page to a bitmap

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```

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

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

// Prepare a bitmap for rendering.
let bitmap = new FSDK.Bitmap(bitmap_width, bitmap_height, FSDK.Bitmap.e_DIBArgb, null, 0);
bitmap.FillRect(0xFFFFFFFF, null);
// Render page.
let render = new FSDK.Renderer(bitmap, false);
render.StartRender(page, matrix, null);
...
```

3.4.2 How to render page and annotation

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Assuming PDFPage page has been loaded and parsed.

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

// Prepare a bitmap for rendering.
let bitmap = new FSDK.Bitmap(bitmap_width, bitmap_height, FSDK.Bitmap.e_DIBArgb, null, 0);
bitmap.FillRect(0xFFFFFFFF, null);

let render = new FSDK.Renderer(bitmap, false);
let dwRenderFlag = FSDK.Renderer.e_RenderAnnot | FSDK.Renderer.e_RenderPage
render.SetRenderContentFlags(dwRenderFlag)
render.StartRender(page, matrix, null)
...
```

3.5 Attachment

In Foxit PDF SDK, attachments are only referred to attachments of documents rather than file attachment annotation, which allow whole files to be encapsulated in a document, much like email attachments. PDF SDK provides applications APIs to access attachments such as loading attachments, getting attachments, inserting/removing attachments, and accessing properties of attachments.

Example:

3.5.1 How to export the embedded attachment file from a PDF and save it as a single file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

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

```

3.5.2 How to remove all the attachments of a PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Assuming PDFDoc doc has been loaded.

// Get information of attachments.
let attachments = new FSDK.Attachments(doc);
let count = attachments.GetCount();
for (var i = 0; i < count; i++) {
    let key = attachments.GetKey(i);
    attachments.RemoveEmbeddedFile(key)
}
...

```

3.6 Text Page

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

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

Example:

3.6.1 How to extract text from a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
const fs = require('fs');
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
let text_page = new FSDK.TextPage(page, FSDK.TextPage.e_ParseTextNormal);
let count = text_page.GetCharCount();
if (count > 0) {
    let text = text_page.GetChars(0, -1);
    let buffer = new Buffer.from(text, 'utf-8');
    fs.writeFileSync(file, buffer, 0, buffer.length);
}
...
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

let rect = new FSDK.RectF();
rect.left = 90;
rect.right = 450;
rect.top = 595;
rect.bottom = 580;
let text_page = new FSDK.TextPage(page, FSDK.TextPage.e_ParseTextNormal);
text_page.GetTextInRect(rect)
...
```

3.7 Text Search

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

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

Example:

3.7.1 How to search a text pattern in a PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```



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

// Search for all pages of doc.
let search = new FSDK.TextSearch(doc, null, FSDK.TextPage.e_ParseTextNormal);

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

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

let flags = FSDK.TextSearch.e_SearchNormal;
search.SetSearchFlags(flags);
...
let match_count = 0;
while (search.FindNext()) {
    let rect_array = search.GetMatchRects();
    OutputMatchedInfo(text_out, search, match_count);
    match_count++;
}
...
```

3.8 Search and Replace

The Search and Replace feature allows you to search for specific text content within a PDF document and replace it with new content.

3.8.1 System requirements

Platform: Windows, Linux, Mac

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

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

SDK Version: Foxit PDF SDK (C, C++, Java, C#, Python, Objective-C) 9.0 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.8.2 How to work with the search and replace function

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

let doc = new FSDK.PDFDoc(input_file);
let error_code = doc.Load("");

// Instantiate a TextSearchReplace object.
let text_searchreplace = new FSDK.TextSearchReplace(doc);
```

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

// Set replacing callback function.
let replace_callback_impl = new ReplaceCallbackImpl();
let replace_callback = new FSDK.ReplaceCallback(replace_callback_impl);
text_searchreplace.SetReplaceCallback(replace_callback);

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

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

3.9 Text Link

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

Example:

3.9.1 How to retrieve hyperlinks in a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

// Get the text page object.
let text_page = new FSDK.TextPage(page);
let pageTextLink = new FSDK.PageTextLinks(text_page);
let textLink = pageTextLink.GetTextLink(index);
let strURL = textLink.GetURI();

...
```

3.10 Bookmark

Foxit PDF SDK provides navigational tools called Bookmarks to allow users to quickly locate and link their point of interest within a PDF document. PDF bookmark is also called outline, and each bookmark contains a destination or actions to describe where it links to. It is a tree-structured hierarchy, so function `FSDK.PDFDoc.GetRootBookmark` must be called first to get the root of the whole bookmark tree before accessing to the bookmark tree. Here, "root bookmark" is an abstract object which can only have some child bookmarks without next sibling bookmarks and any data (includes bookmark data, destination data and action data). It cannot be shown on the application UI since it has no data. Therefore, a root bookmark can only call function `FSDK.Bookmark.GetFirstChild`.

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

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

Example:

3.10.1 How to find and list all bookmarks of a PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Assuming PDFDoc doc has been loaded.
let root = doc.GetRootBookmark();
let first_bookmark = root.GetFirstChild();

function TraverseBookmark(root, iLevel){
  if(!root.IsEmpty()) {
    let child = root.GetFirstChild()
    while(!child.IsEmpty()) {
      TraverseBookmark(child, iLevel + 1)
      child = child.GetNextSibling()
    }
  }
}
...

```

3.10.2 How to insert a new bookmark

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Assuming PDFDoc doc has been loaded.
let root = doc.GetRootBookmark();
if (root.IsEmpty())
{
  root = doc.CreateRootBookmark();
}

let dest = FSDK.Destination.CreateFitPage(doc, i);
let ws_title = `A bookmark to a page (index: ${i})`;
let child = root.Insert(ws_title, FSDK.Bookmark.e_PosLastChild);
child.SetDestination(dest);
child.SetColor(i * 0xF68C21);

```

3.10.3 How to create a table of contents based on bookmark information in PDFs

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

```

```

...
function AddTOCToPDF(doc){
  // Set the table of contents configuration.
  let name_array = new FSDK.Int32Array();
  depth = doc.GetBookmarkLevelDepth()
  if(depth > 0) {
    for(var i = 0; i < depth; i++) {
      name_array.Add(i);
    }
  }

  let title = ""
  let toc_config = new FSDK.TableOfContentsConfig(title, intarray, True, False)
}
// Add the table of contents
doc.AddTableOfContents(toc_config)

```

3.11 Form (AcroForm)

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

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

To import form data from a FDF/XFDF file or export such data to a FDF/XFDF file, please refer to functions [FSDK.PDFDoc.ImportFromFDF](#) and [FSDK.PDFDoc.ExportToFDF](#).

Example:

3.11.1 How to load the forms in a PDF

```

const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.
let hasForm = doc.HasForm()

```

```
if(asForm){
    let form = new FSDK.Form(doc)
}
...
```

3.11.2 How to count form fields and get/set the properties

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

let form = new FSDK.Form();
let count = form.GetFieldCount("");
for (let i = 0; i < count; i++) {
    let field = form.GetField(i, filter);
    let type = field.GetType();
    let org_alternateName = field.GetAlternateName();
    field.SetAlternateName("signature");
}
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Assuming PDFDoc doc has been loaded.
let form = new FSDK.Form();
...
form.ExportToXML(XMLFilePath)
```

3.11.4 How to import form data from a XML file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Assuming PDFDoc doc has been loaded.
let form = new FSDK.Form();
...
form.ImportFromXML(XMLFilePath)
...
```

3.11.5 How to get coordinates of a form field

1. Load PDF file by `FSDK.PDFDoc`.
2. Traverse the `form fields` of the `FSDK.PDFDoc` to get the `field` object of `form`.
3. Traverse the `form controls` of the `field` object to get the `form control` object.
4. Get the related `widget annotation` object by `form control`.
5. Call the `GetRect` of the `widget annotation` object to get the coordinate of the form.

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```

```

...
let doc = new FSDK.PDFDoc("Sample.pdf");
let error_code = doc.Load("");
if (error_code != FSDK.e_ErrSuccess) {
    return 1;
}

if (!doc.HasForm()) return 1;
let form = new FSDK.Form(DOC);
let count = form.GetFieldCount("");
for (let i = 0; i < count; i++) {
    let field = form.GetField(i, "")
    if(field.IsEmpty()) continue
    for (let j = 0; j < field.GetControlCount(); j++) {
        let control = field.GetControl(j)
        let widget = control.GetWidget()
        // Get rectangle of the annot widget.
        let rect = widget.GetRect()
    }
}
...

```

3.12 XFA Form

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

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

Note:

- Foxit PDF SDK provides two callback classes [AppProviderCallback](#) and [DocProviderCallback](#) to represent the callback objects as an XFA document provider and an XFA application provider respectively. All the functions in those classes are used as callback functions. Pure virtual functions should be implemented by users.
- To use the XFA form feature, please make sure the license key has the permission of the 'XFA' module.

Example:

3.12.1 How to load XFADoc and represent an Interactive XFA form

```

const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
let pXFAppHandler = new CFS_XFAppHandler();

```

```
// implement from AppProviderCallback
let app_provider_callback = new FSDK.AppProviderCallback(pXFAAppHandler)
FSDK.Library.RegisterXFAAppProviderCallback(app_provider_callback);
let input_file = input_path + "xfa_dynamic.pdf"
let doc = new FSDK.PDFDoc(input_file);
let error_code = doc.Load("");
if (error_code != FSDK.e_ErrSuccess) {
    return 1;
}

let pXFADocHandler = new CFS_XFADocHandler();
let doc_provider_callback = new FSDK.DocProviderCallback(pXFADocHandler);
let xfa_doc = new FSDK.XFADoc(doc, doc_provider_callback);
xfa_doc.StartLoad(null);
...
```

3.12.2 How to export and import XFA form data

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Assuming FSXFADoc xfa_doc has been loaded.

xfa_doc.ExportData("xfa_form.xml", FSDK.XFADoc.e_ExportDataTypeXML)

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

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

3.13 Form Filler

Form filler is the most commonly used feature for users. Form filler allows applications to fill forms dynamically. The key point for applications to fill forms is to construct some callback functions for PDF SDK to call. To fill the form, please construct a `FSDK.Filler` object by current `FSDK.Form` object or retrieve the `FSDK.Filler` object by function `FSDK.Form.GetFormFiller` if such object has been constructed. (There should be only one form filler object for an interactive form).

3.14 Form Design

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

Example:

3.14.1 How to add a text form field to a PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Assuming PDFDoc doc has been loaded.
// Add text field
control = form.AddControl(page, "Text Field0", FSDK.Field.e_TypeTextField, new FSDK.RectF(50, 600, 90, 640))
control.GetField().SetValue("3")
// Update text field's appearance.
control.GetWidget().ResetAppearanceStream()

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

...
```

3.14.2 How to remove a text form field from a PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Assuming PDFDoc doc has been loaded.
let form = new FSDK.Form(doc)
let filter = "text1"
countFields = form.GetFieldCount("")
for(var i = 0; i < countFields; i++) {
    let field = form.GetField(i, filter)
    if(field.GetType() == FSDK.Field.e_TypeTextField)
        form.RemoveField(field)
}

...
```

3.15 Annotations

3.15.1 General

An annotation associates an object such as note, line, and highlight with a location on a page of a PDF document. It provides a way to interact with users by means of the mouse and keyboard. PDF includes a wide variety of standard annotation types as listed in Table 3-1. Among these annotation types, many of them are defined as markup annotations for they are used primarily to mark up PDF documents. These annotations have text that appears as part of the annotation and may be displayed in other ways by a conforming reader, such as in a Comments pane. The 'Markup' column in Table 3-1 shows whether an annotation is a markup annotation.

Foxit PDF SDK supports most annotation types defined in PDF reference ^[1]. PDF SDK provides APIs of annotation creation, properties access and modification, appearance setting and drawing.

Table 3-1

Annotation type	Description	Markup	Supported by SDK
Text(Note)	Text annotation	Yes	Yes
Link	Link Annotation	No	Yes
FreeText (TypeWriter/TextBox/Callout)	Free text annotation	Yes	Yes
Line	Line annotation	Yes	Yes
Square	Square annotation	Yes	Yes
Circle	Circle annotation	Yes	Yes
Polygon	Polygon annotation	Yes	Yes
PolyLine	PolyLine annotation	Yes	Yes
Highlight	Highlight annotation	Yes	Yes
Underline	Underline annotation	Yes	Yes
Squiggly	Squiggly annotation	Yes	Yes
StrikeOut	StrikeOut annotation	Yes	Yes
Stamp	Stamp annotation	Yes	Yes
Caret	Caret annotation	Yes	Yes
Ink(pencil)	Ink annotation	Yes	Yes
Popup	Popup annotation	No	Yes
File Attachment	FileAttachment annotation	Yes	Yes
Sound	Sound annotation	Yes	No
Movie	Movie annotation	No	No
Widget*	Widget annotation	No	Yes
Screen	Screen annotation	No	Yes
PrinterMark	PrinterMark annotation	No	No
TrapNet	Trap network annotation	No	No
Watermark*	Watermark annotation	No	Yes
3D	3D annotation	No	No
Redact	Redact annotation	Yes	Yes

Note:

1. The annotation types of widget and watermark are special. They aren't supported in the module of 'Annotation'. The type of widget is only used in the module of 'form filler' and the type of watermark only in the module of 'watermark'.
2. Foxit SDK supports a customized annotation type called PSI (pressure sensitive ink) annotation that is not described in PDF reference [1]. Usually, PSI is for handwriting features and Foxit SDK treats it as PSI annotation so that it can be handled by other PDF products.

Example:

3.15.1.1 How to add a link annotation to a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...

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

// Add link annotation.
let link = new FSDK.Link(page.AddAnnot(FSDK.Annot.e_Link, new FSDK.RectF(350,350,380,400)));
link.SetHighlightingMode(FSDK.Annot.e_HighlightingToggle);
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...

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

// Add highlight annotation.
let highlight= new FSDK.Highlight(page.AddAnnot(FSDK.Annot.e_Highlight, new FSDK.RectF(10,450,100,550)));
// This flag is used for printing annotations.
highlight.SetFlags(4);
highlight.SetContent("Highlight");
let quad_points = new FSDK.QuadPoints();
quad_points.first = new FSDK.PointF(10, 500);
quad_points.second = new FSDK.PointF(90, 500);
quad_points.third = new FSDK.PointF(10, 480);
quad_points.fourth = new FSDK.PointF(90, 480);
let quad_points_array = new FSDK.QuadPointsArray();
quad_points_array.Add(quad_points);
highlight.SetQuadPoints(quad_points_array);
highlight.SetSubject("Highlight");
highlight.SetTitle("Foxit SDK");
highlight.SetCreationDateTime(GetLocalDateTime());
highlight.SetModifiedDateTime(GetLocalDateTime());
highlight.SetUniqueID(RandomUID());
// Appearance should be reset.
highlight.ResetAppearanceStream();
```

3.15.1.3 How to set the popup information when creating markup annotations

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...

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

// Create a new note annot and set the properties for it.
let note = new FSDK.Note(page.AddAnnot(FSDK.Annot.e_Note, new FSDK.RectF(10,350,50,400)));
// This flag is used for printing annotations.
```

```
note.SetFlags(4);
note.SetIconName("Comment");
note.SetSubject("Note");
note.SetTitle("Foxit SDK");
note.SetContent("Note annotation.");
note.SetCreationDateTime(GetLocalDateTime());
note.SetModifiedDateTime(GetLocalDateTime());
note.SetUniqueID(RandomUID());
// Add popup to note annotation
let popup = new FSDK.Popup(page.AddAnnot(FSDK.Annot.e_Popup, new FSDK.RectF(300,450,500,550)));
popup.SetBorderColor(0x00FF00);
popup.SetOpenStatus(false);
popup.SetModifiedDateTime(GetLocalDateTime());
note.SetPopup(popup);
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.
...

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

// Get page transformation matrix.
let displayMatrix= page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());
let iAnnotCount = page.GetAnnotCount();
for(var i = 0; i < iAnnotCount; i++) {
    let pAnnot = page.GetAnnot(i)
    if(FSDK.Annot.e_Popup == pAnnot.GetType()) continue;
    let annotRect = pAnnot.GetDeviceRect(False, displayMatrix);
    let pt = new FSDK.PointF();
    let tolerance = 1.0;

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

3.15.1.5 How to extract the texts under text markup annotations

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.
...

let page = doc.GetPage(0)
// Parse the first page.
page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false)
```

```
let annot_count = page.GetAnnotCount()
let text_page = new FSDK.TextPage(page)
for(var i = 0; i < annot_count; i++) {
    let annot = page.GetAnnot(i)
    let text_markup = TextMarkup(annot)
    if(text_markup.IsEmpty()) {
        // Get the texts which intersect with a text markup annotation.
        text = text_page.GetTextUnderAnnot(text_markup)
    }
}
```

3.15.1.6 How to add richtext for freetext annotation

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// 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.
let freetext = new FSDK.FreeText(page.AddAnnot(FSDK.Annot.e_FreeText, new FSDK.RectF(450, 50, 550, 100)));
// Set annotation's properties.

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

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

richtext_style.font = new FSDK.Font("Calibri", 0, FSDK.Font.e_CharsetANSI, 0);

richtext_style.text_color = 0x0000FF;
richtext_style.is_underline = false;
richtext_style.is_strikethrough = true;
richtext_count = freetext.GetRichTextCount();
freetext.InsertRichText(richtext_count - 1, "2_strikethrough ", richtext_style);

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

3.15.2 Import annotations from or export annotations to a FDF file

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

Example:

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

```
fdf_doc = new FSDK.FDFDoc(buffer, file_size)
pdf_doc.ImportFromFDF(fdf_doc, FSDK.PDFDoc.e_Annots)
```

3.16 Image Conversion

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

Example:

3.16.1 How to convert PDF pages to bitmap files

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

// Get page count
let page_count = doc.GetPageCount();
for(let i=0; i<page_count; i++) {
    let page = doc.GetPage(i);
    // Parse page.
    page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false);
    let width = page.GetWidth();
    let height = page.GetHeight();
    let matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

    // Prepare a bitmap for rendering.
    let bitmap = new FSDK.Bitmap(width, height, FSDK.Bitmap.e_DIBArgb, null, 0);
    bitmap.FillRect(0xFFFFFFFF, null);

    // Render page
    let render = new FSDK.Renderer(bitmap, false);
    render.StartRender(page, matrix, null);
    image.AddFrame(bitmap);
    ...
}
```

Note: For pdf2image functionality, if the PDF file contains images larger than 1G, it is recommended to process the images using tiled rendering. Otherwise, it may occur exceptions. Following is a brief implementation of tiled rendering.

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```

```
...
// Parse page.
page.StartParse(PDFPage.e_ParsePageNormal, null, false);

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

let render_sum = 10;
let width_scale = 1;
let height_scale = 1;
let little_width = width * width_scale;
let little_height = height / render_sum * height_scale;
for(var i = 0; i < render_sum; i++) {
    // According to Matrix, do module rendering for large PDF files.
    let matrix = page.GetDisplayMatrix(0, -1 * i * little_height, little_width, height * height_scale,
page.GetRotation());
    // Prepare a bitmap for rendering.
    let bitmap = new FSDK.Bitmap(little_width, little_height, FSDK.Bitmap.e_DIBArgb)
    bitmap.FillRect(0xFFFFFFFF, null);
    let render = new FSDK.Renderer(bitmap, false);
    render.StartRender(page, matrix, null);
    // The bitmap data will be added to the end of image file after rendering.
}
...
```

3.16.2 How to convert an image file to PDF file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

doc = PDFDoc()
for(var i = 0; i < count; i++) {
    page = doc.InsertPage(i)
    page.StartParse(FSDK.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, FSDK.PDFDoc.e_SaveFlagNoOriginal)
...
```

3.17 Watermark

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

// Assuming PDFDoc doc has been loaded.

let settings = new FSDK.WatermarkSettings();
settings.flags = FSDK.WatermarkSettings.e_FlagASPageContents | FSDK.WatermarkSettings.e_FlagOnTop;
settings.offset_x = 0;
settings.offset_y = 0;
settings.opacity = 90;
settings.position = FSDK.e_PosTopRight;
settings.rotation = -45.0;
settings.scale_x = 1.0;
settings.scale_y = 1.0;

let text_properties = new FSDK.WatermarkTextProperties();
text_properties.alignment = FSDK.e_AlignmentCenter;
text_properties.color = 0xF68C21;
text_properties.font_style = FSDK.WatermarkTextProperties.e_FontStyleNormal;
text_properties.line_space = 1;
text_properties.font_size = 12.0;
text_properties.font = new FSDK.Font(FSDK.Font.e_StdIDTimesB);

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

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

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

// Assuming PDFDoc doc has been loaded.

let settings = new FSDK.WatermarkSettings();
settings.flags = FSDK.WatermarkSettings.e_FlagASPageContents | FSDK.WatermarkSettings.e_FlagOnTop;
settings.offset_x = 0.0;
settings.offset_y = 0.0;
settings.opacity = 20;
settings.position = FSDK.e_PosCenter;
settings.rotation = 0.0;

let image = new FSDK.Image(image_file);
let bitmap = image.GetFrameBitmap(0);
settings.scale_x = page.GetWidth() * 0.618 / bitmap.GetWidth();
```

```
settings.scale_y = settings.scale_x;

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

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

3.17.3 How to remove all watermarks from a page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFPage page has been loaded and parsed.
...
page.RemoveAllWatermarks()
...
// Save document to file.
```

3.18 Barcode

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

Table 3-2

Barcode Type	Code39	Code128	EAN 8	UPC A	EAN13	ITF	PDF417	QR
Dimension	1D	1D	1D	1D	1D	1D	2D	2D

Example:

3.18.1 How to generate a barcode bitmap from a string

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

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

// Barcode format types.
```



```
let code_format = FSDK.Barcode.e_FormatCode39;

// Format error correction level of QR code.
let sz_qr_level = FSDK.Barcode.e_QRCorrectionLevelLow;

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

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

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

let barcode = new FSDK.Barcode();
let bitmap = barcode.GenerateBitmap(sz_code_string, code_format, unit_width, unit_height, sz_qr_level);
```

3.19 Security

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

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

Example:

3.19.1 How to encrypt a PDF file with Foxit DRM

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

let doc = new FSDK.PDFDoc(input_file);
let error_code = doc.Load("");
if (error_code != FSDK.e_ErrSuccess) {
    console.log("The Doc [%s] Error: %d\n", input_file, error_code);
    return false;
}

// Do encryption.
let handler = new FSDK.DRMSecurityHandler();
let file_id = "Simple-DRM-file-ID";
let initialize_key = "Simple-DRM-initialize-key";
let encrypt_data = new FSDK.DRMEncryptData(true, "Simple-DRM-filter", FSDK.SecurityHandler.e_CipherAES, 16,
true, 0xffffffffc);
```

```
handler.Initialize(encrypt_data, file_id, initialize_key);
doc.SetSecurityHandler(handler);

let output_file = output_directory + "foxit_drm_encrypt.pdf";
doc.SaveAs(output_file, FSDK.PDFDoc.e_SaveFlagNoOriginal);
```

3.20 Reflow

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

Example:

3.20.1 How to create a reflow page and render it to a bmp file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

// Assuming PDFDoc doc has been loaded.

let page = doc.GetPage(0);
// Parse PDF page.
page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false);

let reflow_page = new FSDK.ReflowPage(page);

// Set some arguments used for parsing the reflow page.
reflow_page.SetLineSpace(0);
reflow_page.SetZoom(100);
reflow_page.SetParseFlags(FSDK.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.
let content_width = reflow_page.GetContentWidth();
let content_height = reflow_page.GetContentHeight();

// Assuming Bitmap bitmap has been created.

// Render reflow page.
let renderer = new FSDK.Renderer(bitmap, false);
let matrix = reflow_page.GetDisplayMatrix(0, 0, content_width, content_height, FSDK.e_Rotation0);
renderer.StartRenderReflowPage(reflow_page, matrix, null);
```

3.21 Asynchronous PDF

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

3.22 Pressure Sensitive Ink

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
let psi = new FSDK.PSI(480, 180, true);

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

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

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

// Add points to pressure sensitive ink.
let x = 121.3043
let y = 326.6846
let pressure = 0.0966
let type = FSDK.Path.e_TypeMoveTo
psi.AddPoint(new FSDK.PointF(x, y), type, pressure)
```

3.23 Wrapper

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

Example:

3.23.1 How to open a document including wrapper data

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
// file_name is PDF document which includes wrapper data.
let doc = new FSDK.PDFDoc(file_name);
let code = doc.Load("");
if (code !== FSDK.e_ErrSuccess) {
    return false;
}

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

let offset = doc.GetWrapperOffset();

let file_reader = new FileReader(offset);
file_reader.LoadFile(file_name);
let reader_callback = new FSDK.FileReaderCallback(file_reader);
...
```

3.24 PDF Objects

There are eight types of objects in PDF: Boolean object, numerical object, string object, name object, array object, dictionary object, stream object and null object. PDF objects are document level objects that are different from page objects (see 3.25) which are associated with a specific page each. Foxit PDF SDK provides APIs to create, modify, retrieve and delete these objects in a document.

Example:

3.24.1 How to remove some properties from catalog dictionary

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
```

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

let catalog = doc.GetCatalog();
if (null == catalog) return;

let key_strings = [ "Type", "Boolean", "Name", "String", "Array", "Dict"];

let count = key_strings.length;
for (let i = 0; i < count; i++) {
    if (catalog.HasKey(key_strings[i]))
        catalog.RemoveAt(key_strings[i]);
}
...
```

3.25 Page Object

Page object is a feature that allows novice users having limited knowledge of PDF objects (see 3.24 for details of PDF objects) to be able to work with text, path, image, and canvas objects. Foxit PDF SDK provides APIs to add and delete PDF objects in a page and set specific attributes. Using page object, users can create PDF page from object contents. Other possible usages of page object include adding headers and footers to PDF documents, adding an image logo to each page, or generating a template PDF on demand.

Example:

3.25.1 How to create a text object in a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

// Assuming PDFPage page has been loaded and parsed.

let position = page.GetLastGraphicsObjectPosition(FSDK.GraphicsObject.e_TypeText);
let text_object = FSDK.TextObject.Create();

text_object.SetFillColor(0xFFFF7F00);

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

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

3.25.2 How to add an image logo to a PDF page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
// Assuming PDFPage page has been loaded and parsed.

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

let width = image.GetWidth();
let height = image.GetHeight();

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

// Please notice the matrix value.
image_object.SetMatrix(new FSDK.Matrix2D(width, 0, 0, height, (page_width - width) / 2.0, (page_height - height) / 2.0));

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

3.26 Marked content

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFPage page has been loaded and parsed.

let position = page.GetLastGraphicsObjectPosition(FSDK.GraphicsObject.e_TypePath);
let text_obj = page.GetGraphicsObject(position).GetTextObject();
let content = text_obj.GetMarkedContent();
let item_count = content.GetItemCount();

// Get marked content property.
for (let i = 0; i < item_count; i++) {
    text_doc.Write("index: %d\r\n", i);
    let tag_name = content.GetItemTagName(i);
}
```

```
let mcid = content.GetItemMCID(i);
}
...
```

3.27 Layer

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

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

Example:

3.27.1 How to create a PDF layer

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.

let layertree = new FSDK.LayerTree(doc);
let root = layertree.GetRootNode();
if (root.IsEmpty()) {
    console.log("No layer information!\r\n");
    return;
}
...
```

3.27.2 How to set all the layer nodes information

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.

function SetAllLayerNodesInformation(layer_node) {
    if (layer_node.HasLayer()) {
        layer_node.SetDefaultVisible(true);
        layer_node.SetExportUsage(FSDK.LayerTree.e_StateUndefined);
        layer_node.SetViewUsage(FSDK.LayerTree.e_StateOFF);
        let print_data = new FSDK.LayerPrintData("subtype_print", FSDK.LayerTree.e_StateON);
        layer_node.SetPrintUsage(print_data);
        let zoom_data = new FSDK.LayerZoomData(1, 10);
    }
}
```

```

layer_node.SetZoomUsage(zoom_data);
let new_name = "[View_OFF_Print_ON_Export_Undefined]" + layer_node.GetName();
layer_node.SetName(new_name);
}
let count = layer_node.GetChildrenCount();
for (let i = 0; i < count; i++) {
    let child = layer_node.GetChild(i);
    SetAllLayerNodesInformation(child);
}
}
}

let layertree = new FSDK.LayerTree(doc);
let root = layertree.GetRootNode();
SetAllLayerNodesInformation(root);
...

```

3.27.3 How to edit layer tree

```

const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// Assuming PDFDoc doc has been loaded.

// edit layer tree.
let doc = new FSDK.PDFDoc(input_file);
let error_code = doc.Load("");
let layertree = new FSDK.LayerTree(doc);
let root = layertree.GetRootNode();
let children_count = root.GetChildrenCount();
root.RemoveChild(children_count - 1);
let child = root.GetChild(children_count - 2);
let child0 = root.GetChild(0);
child.MoveTo(child0, 0);
child.AddChild(0, "AddedLayerNode", true);
child.AddChild(0, "AddedNode", false);

```

3.28 Signature

PDF Signature module can be used to create and sign digital signatures for PDF documents, which protects the security of documents' contents and avoids it to be tampered maliciously. It can let the receiver make sure that the document is released by the signer and the contents of the document are complete and unchanged. Foxit PDF SDK provides APIs to create digital signature, verify the validity of signature, delete existing digital signature, get and set properties of digital signature, display signature and customize the appearance of the signature form fields.

Note: Foxit PDF SDK provides default Signature callbacks which supports the following two types of signature filter and subfilter:

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

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

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

Example:

3.28.1 How to sign the PDF document with a signature

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
let filter = "Adobe.PPKLite";
let sub_filter = "adbe.pkcs7.detached";
console.log("Use signature callback object for filter \"%s\" and sub-filter \"%s\"", filter, sub_filter);
let pdf_page = pdf_doc.GetPage(0);
// Add a new signature to first page.
let new_signature = AddSignature(pdf_page, sub_filter);
// Set filter and subfilter for the new signature.
new_signature.SetFilter(filter);
new_signature.SetSubFilter(sub_filter);
let is_signed = new_signature.IsSigned();
let sig_state = new_signature.GetState();
console.log("[Before signing] Signed?:%s\\t State:%s", is_signed? "true" : "false",
TransformSignatureStateToString(sig_state));
// Sign the new signature.
let signed_pdf_path = output_directory + "signed_newsignature_default_handler.pdf";

let cert_file_path = input_path + "foxit_all.pfx";
let cert_file_password = "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(cert_file_path, cert_file_password,
    FSDK.Signature.e_DigestSHA1, signed_pdf_path, null, null);
console.log("[Sign] Finished!");
is_signed = new_signature.IsSigned();
sig_state = new_signature.GetState();
console.log("[After signing] Signed?:%s\\t State:%s", is_signed? "true" : "false",
TransformSignatureStateToString(sig_state));

// Open the signed document and verify the newly added signature (which is the last one).
console.log("Signed PDF file: %s", signed_pdf_path);
let signed_pdf_doc = new FSDK.PDFDoc(signed_pdf_path);
error_code = signed_pdf_doc.Load("");
if (FSDK.e_ErrSuccess != error_code) {
    console.log("Fail to open the signed PDF file.");
    return;
}
// Get the last signature which is just added and signed.
let sig_count = signed_pdf_doc.GetSignatureCount();
let signed_signature = signed_pdf_doc.GetSignature(sig_count-1);
// Verify the integrity of signature.
```

```
signed_signature.StartVerify(Buffer.alloc(0), null);
console.log("[Verify] Finished!");
is_signed = signed_signature.IsSigned();
sig_state = signed_signature.GetState();
console.log("[After verifying] Signed?:%s\\tState:%s", is_signed? "true" : "false",
TransformSignatureStateToString(sig_state));
```

3.29 PAdES

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

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

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

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

3.30 PDF Action

PDF Action is represented as the base PDF action class. Foxit PDF SDK provides APIs to create a series of actions and get the action handlers, such as embedded goto action, JavaScript action, named action and launch action, etc.

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
```

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

// Add link annotation.
let link = new FSDK.Link(page.AddAnnot(FSDK.Annot.e_Link, new FSDK.RectF(350,350,380,400)));
// This flag is used for printing annotations.
link.SetFlags(4);
link.SetHighlightingMode(FSDK.Annot.e_HighlightingPush);

// Add action for link annotation.
let action = new FSDK.URIAction(FSDK.Action.Create(page.GetDocument(), FSDK.Action.e_TypeURI));
action.SetTrackPositionFlag(true);
action.SetURI("www.foxitsoftware.com");
link.SetAction(action);
// Appearance should be reset.
link.ResetAppearanceStream();
```

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

// Add link annotation
let link = new FSDK.Link(page.AddAnnot(FSDK.Annot.e_Link, new FSDK.RectF(350,350,380,400)));
link.SetHighlightingMode(FSDK.Annot.e_HighlightingToggle);

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

3.31 JavaScript

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

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

The JavaScript methods and properties supported by Foxit PDF SDK are listed in the [appendix](#).

Example:

3.31.1 How to add JavaScript Action to Document

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

javascript_action = new FSDK.JavaScriptAction(FSDK.Action.Create(form.GetDocument(),
FSDK.Action.e_TypeJavaScript))
javascript_action.SetScript("app.alert(\"Hello Foxit \");");
additional_act = AdditionalAction(doc);
additional_act.SetAction(FSDK.AdditionalAction.e_TriggerDocWillClose,javascript_action);
additional_act.DoJSAction(FSDK.AdditionalAction.e_TriggerDocWillClose);
...
```

3.31.2 How to add JavaScript Action to Annotation

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

javascript_action = new FSDK.JavaScriptAction(FSDK.Action.Create(form.GetDocument(),
FSDK.Action.e_TypeJavaScript))
javascript_action.SetScript("app.alert(\"Hello Foxit \");");
additional_act = AdditionalAction(annot);
additional_act.SetAction(FSDK.AdditionalAction.e_TriggerAnnotMouseButtonPressed, javascript_action);
additional_act.DoJSAction(FSDK.AdditionalAction.e_TriggerAnnotMouseButtonPressed);
...
```

3.31.3 How to add JavaScript Action to FormField

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

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

control1 = form.AddControl(page, "Text Field1", FSDK.Field.e_TypeTextField, new FSDK.RectF(100, 600, 140, 640));
control1.GetField().SetValue("23");
```

```
// Update text field's appearance.
control1.GetWidget().ResetAppearanceStream();

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

3.31.4 How to add a new annotation to PDF using JavaScript

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

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

...
```

3.31.5 How to get/set properties of annotations (strokeColor, fillColor, readOnly, rect, type) using JavaScript

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

// Get properties of annotations.
let javascript_action = new FSDK.JavaScriptAction(Action.Create(form.GetDocument(),
FSDK.Action.e_TypeJavaScript));
javascript_action.SetScript("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);});");
additional_act = AdditionalAction(field);
additional_act.SetAction(FSDK.AdditionalAction.e_TriggerAnnotCursorEnter,javascript_action);
additional_act.DoJSAction(FSDK.AdditionalAction.e_TriggerAnnotCursorEnter);
```

```
// Set properties of annotations (only take strokeColor as an example).
let javascript_action1 = new FSDK.JavaScriptAction(FSDK.Action.Create(form.GetDocument(),
FSDK.Action.e_TypeJavaScript));
javascript_action1.SetScript("var ann = this.getAnnot(0, \"UniqueID\");if (ann != null) { ann.strokeColor =
color.blue; }");
let additional_act1 = AdditionalAction(field1);
additional_act1.SetAction(FSDK.AdditionalAction.e_TriggerAnnotCursorEnter,javascript_action1);
additional_act1.DoJSAction(FSDK.AdditionalAction.e_TriggerAnnotCursorEnter);
...
```

3.31.6 How to destroy annotation using JavaScript

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

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

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

3.32 Redaction

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

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

To do Redaction, you can use the following APIs:

- Call function [FSDK.Redaction.Redaction](#) to create a redaction module. If module "Redaction" is not defined in the license information which is used in function [FSDK.Library.Initialize](#), it means user has no right in using redaction related functions and this constructor will throw exception [FSDK.e_ErrInvalidLicense](#).
- Then call function [FSDK.Redaction.MarkRedactAnnot](#) to create a redaction object and mark page contents (text object, image object, and path object) which are to be redacted.

- Finally call function `FSDK.Redaction.Apply` to apply redaction in marked areas: remove the text or graphics under marked areas permanently.

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let redaction = new FSDK.Redaction(doc);
// Parse PDF page.
let page = doc.GetPage(0);
page.StartParse(FSDK.PDFPage.e_ParsePageNormal, null, false);
let text_page = new FSDK.TextPage(page, FSDK.TextPage.e_ParseTextNormal);
let text_search = new FSDK.TextSearch(text_page);
text_search.SetPattern("PDF");
let rect_array = new FSDK.RectFArray();
while(text_search.FindNext()) {
    itemArray = text_search.GetMatchRects()
    rect_array.InsertAt(rect_array.GetSize(), itemArray)
}
if(rect_array.GetSize() > 0) {
    let redact = redaction.MarkRedactAnnot(page, rect_array);
    redact.ResetAppearanceStream();
    doc.SaveAs(output_directory + "AboutFoxit_redacted_default.pdf", FSDK.PDFDoc.e_SaveFlagNormal);

    // Set border color to green.
    redact.SetBorderColor(0x00FF00);
    // Set fill color to blue.
    redact.SetFillColor(0x0000FF);
    // Set rollover fill color to red.
    redact.SetApplyFillColor(0xFF0000);
    redact.ResetAppearanceStream();
    doc.SaveAs(output_directory + "AboutFoxit_redacted_setColor.pdf", FSDK.PDFDoc.e_SaveFlagNormal);

    redact.SetOpacity(0.5);
    redact.ResetAppearanceStream();
    doc.SaveAs(output_directory + "AboutFoxit_redacted_setOpacity.pdf", FSDK.PDFDoc.e_SaveFlagNormal);

    if(redaction.Apply())
        console.log("Redact page(0) succeed.");
    else
        console.log("Redact page(0) succeed.");
}
doc.SaveAs(output_directory + "AboutFoxit_redacted_apply.pdf", FSDK.PDFDoc.e_SaveFlagNormal);
```

3.33 Comparison

Comparison feature lets you see the differences in two versions of a PDF. Foxit PDF SDK provides APIs to compare two PDF documents page by page, the differences between the two documents will be returned.

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

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let base_doc = new FSDK.PDFDoc(input_base_file);
let error_code = base_doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    console.log("The Doc [%s] Error: %d\n", input_base_file, error_code);
    return 1;
}

let compared_doc = new FSDK.PDFDoc(input_compared_file);
error_code = compared_doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    console.log("The Doc [%s] Error: %d\n", input_compared_file, error_code);
    return 1;
}

let comparison = new FSDK.Comparison(base_doc, compared_doc);
let result = comparison.DoCompare(0, 0, FSDK.Comparison.e_CompareTypeText);
let old_info = result.base_doc_results;
let new_info = result.compared_doc_results;
let = result.compared_doc_results;
let old_info_size = old_info.GetSize();
let new_info_size = new_info.GetSize();
let page_base = base_doc.GetPage(0);
let page = compared_doc.GetPage(0);
for (let i=0; i<new_info_size; i++) {
    let item = new_info.GetAt(i);
    let type = item.type;
    if (type === FSDK.CompareResultInfo.e_CompareResultTypeDeleteText) {
        let res_string = `\"${item.diff_contents}\"`;
        CreateDeleteTextStamp(page, item.rect_array, 0xff0000, res_string, "Compare : Delete", "Text");
    }
}
```



```

} else if (type == FSDK.CompareResultInfo.e_CompareResultTypeInsertText) {
    let res_string = `\"${item.diff_contents}\"`;
    CreateDeleteText(page, item.rect_array, 0x0000ff, res_string, "Compare : Insert", "Text");
} else if (type == FSDK.CompareResultInfo.e_CompareResultTypeReplaceText) {
    let res_string = `[New]: \"${new_info.GetAt(i).diff_contents}\"\\r\\n[Old]: \"${item.diff_contents}\"`;
    CreateSquigglyRect(page, item.rect_array, 0xe7651a, res_string, "Compare : Replace", "Text");
}
}
// Save the comparison result to a PDF file.
compared_doc.SaveAs(output_directory + "new.pdf", FSDK.PDFDoc.e_SaveFlagNormal);

```

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

3.34 OCR

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

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

3.34.1 System requirements

Platform: Windows, Linux (x64)

Programming Language: C, C++, Java, Python, C#, Node.js

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

SDK Version: Foxit PDF SDK for Windows (C++, Java, C#) 6.4 or higher; Foxit PDF SDK (C) 7.4 or higher; Foxit PDF SDK for Windows (Python) 8.3 or higher; Foxit PDF SDK for Linux x64 (C++, Java, C#, Python) 9.0 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

Note:

For Linux platform, in some cases, particularly within a clean Docker environment, you may encounter an engine initialization failure due to the lack of certain libraries during the container setup. The '[libFREngine.so](#)' in the engine may lack '[libgomp.so.1](#)', which can cause this issue.

To resolve this issue, perform the following commands in Docker:

```

sudo apt-get update
sudo apt-get install libgomp1

```

3.34.2 Trial limit for SDK OCR add-on module

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

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

3.34.3 OCR resource files

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

For Windows:

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

- **debugging_files:** Resource files used for debugging the OCR project. These file(s) cannot be distributed.
- **language_resource_CJK:** Resource files for CJK language, including: Chinese-Simplified, Chinese-Traditional, Japanese, and Korean.
- **language_resources_noCJK:** Resource files for the languages except CJK, including: Basque, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Faeroese, Finnish, French, Galician, German, Greek, Hebrew, Hungarian, Icelandic, Italian, Latvian (Lettish), Lithuanian, Macedonian, Maltese, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swedish, Thai, Turkish, Ukrainian.
- **win32_lib:** 32-bit library resource files
- **win64_lib:** 64-bit library resource files
- **readme.txt:** A txt file for introducing the role of each folder in this directory, as well as how to use those resource files for OCR.

For Linux x64:

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

- **Data:** Data and resource files for following languages:
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:** Library files for Linux x64.

3.34.4 How to run the OCR demo

Foxit PDF SDK for Node.js (Windows and Linux x64) provides an OCR demo located in the "`\examples\simple_demo\ocr`" folder to show you how to use Foxit PDF SDK to do OCR for a PDF page or a PDF document.

3.34.4.1 Build an OCR resource directory

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

Note: Starting from version 10.1, Foxit PDF SDK has restructured the **FSDK.OCREngine.Initialize** interface and added a new parameter, **is_shared_cpu_cores_mode**, to specify whether to use multi-process mode. When this parameter value is true, multi-process mode will be used during the OCR process, and when the value is false, single-process mode will be used.

For Windows:

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

- 1) Create a new folder to add the resources. For example, "`D:/ocr_resources`".
- 2) Add the appropriate library resource based on the platform architecture.
 - For **win32**, copy **all the files** under "`ocr_addon/win32_lib`" folder to "`D:/ocr_resources`".
 - For **win64**, copy **all the files** under "`ocr_addon/win64_lib`" folder to "`D:/ocr_resources`".
- 3) Add the language resource.
 - For CJK (Chinese-Simplified, Chinese-Traditional, Japanese, and Korean), copy **all the files** under "`ocr_addon/language_resource_CJK`" folder to "`D:/ocr_resources`".

- For all other languages except CJK, copy **all the files** under "ocr_addon/language_resources_noCJK" folder to "D:/ocr_resources".
- For all the supported languages, copy **all the files** under "ocr_addon/language_resource_CJK" and "ocr_addon/language_resources_noCJK" folders to "D:/ocr_resources".

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

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

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

For Linux x64:

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

- 1) Create a new folder to add the resources. For example, "/root/Desktop/ocr_resources".
- 2) Copy the whole folders of "**Data**", "**Bin**" under the "ocr_addon_linux" to "/root/Desktop/ocr_resources".

Then, the OCR resource files path is set to "**/root/Desktop/ocr_resources/Bin**".

Note: Before loading the resource files, please set the environment variable for loading the library, please execute: `export LD_LIBRARY_PATH=/root/Desktop/ocr_resources/Bin`.

3.34.4.2 Configure the demo

After building the OCR resource directory, configure the demo in the "\examples\simple_demo\ocr\ocr.js" file. Following will configure the demo in "ocr.js" file on Windows for example. For Linux x64 platform, do the similar configuration with Windows.

Specify the OCR resource directory

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

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

Choose the language resource

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

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

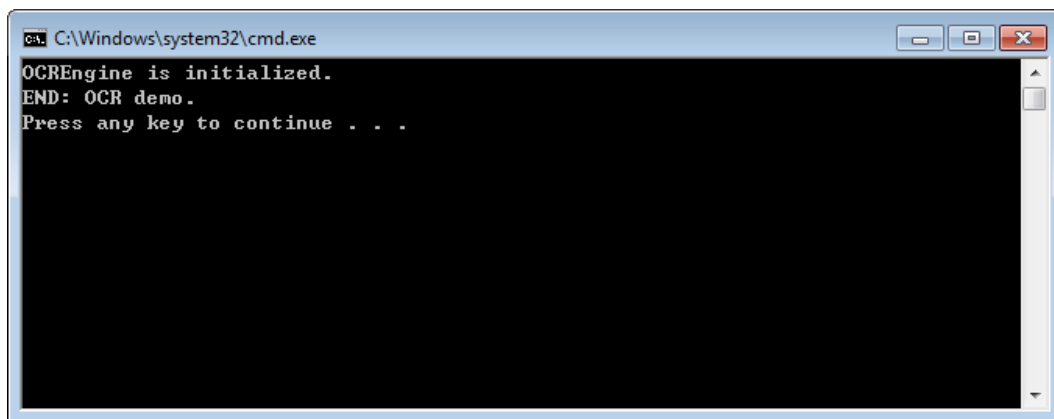
(Optional) Set log for OCREngine

If you want to print the entire log of the OCR Engine, please uncomment the **FSDK.OCREngine.SetLogFile** method as below:

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

3.34.4.3 Run the demo

Once you run the demo successfully by "node ocr.js" in the CMD, the console will print the following by default:



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

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

3.35 Compliance

PDF Compliance

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

PDF/A Compliance

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

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

PDF/E Compliance

PDF/E is an ISO-standardized version of the PDF specialized for the reliable exchange and archiving of engineering documents. It is designed for the creation, exchange, archiving, and printing documents used in engineering workflows.

From version 10.1, Foxit PDF SDK provides APIs to convert a PDF to be compliance with PDF/E standard or verify whether a PDF is compliance with PDF/E standard. It supports the PDF/E-1 version.

PDF/X Compliance

PDF/X is an ISO-standardized version of the PDF specialized for the exchange of graphics-intensive documents. It is mainly used to ensure consistency and predictability when printing files in fields such as design, drawing, engineering, and graphic arts.

From version 10.1, Foxit PDF SDK provides APIs to convert a PDF to be compliance with PDF/X standard or verify whether a PDF is compliance with PDF/X standard. It supports the PDF/X version including PDF/X-1a, PDF/A-3, PDF/A-4, PDF/A-4p.

Preflight Feature

From version 10.1, Foxit PDF SDK supports preflight feature, which allows users to utilize Foxit PDF SDK to get preflight keys and analyze or fixup PDF file.

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

3.35.1 System requirements

Platform: Windows, Linux (x86 and x64), Mac

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

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

SDK Version: Foxit PDF SDK (C++, Java, C#, Objective-C) 6.4 or higher (for PDF Compliance, it requires Foxit PDF SDK 7.1 or higher); Foxit PDF SDK (C) 7.4 or higher; Foxit PDF SDK (Python) 8.3 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

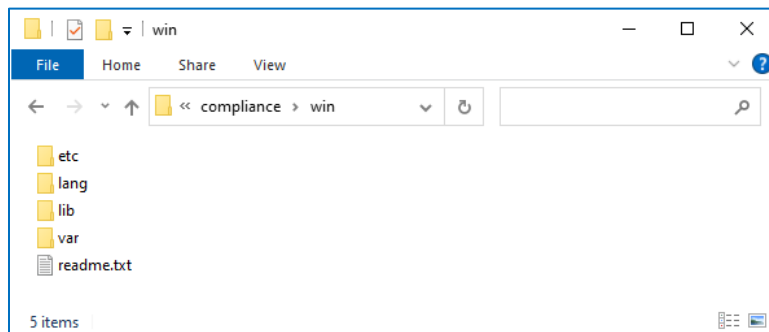
Note: For PDF/E, PDF/X, and preflight feature, it requires Foxit PDF SDK 10.1.

3.35.2 Compliance resource files

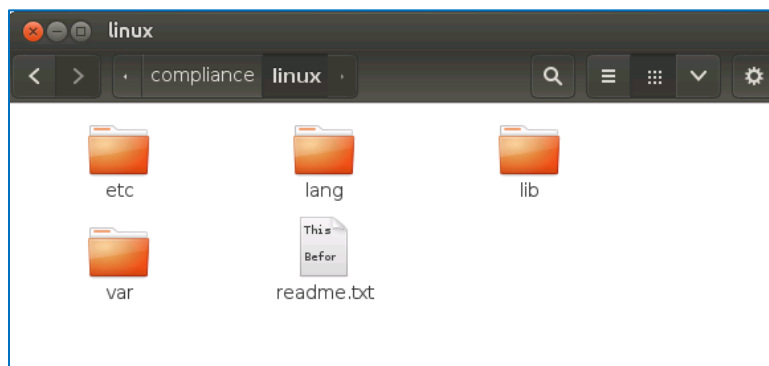
Please contact Foxit support team or sales team to get the Compliance resource files package.

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

For **Windows**:



For **Linux**:



3.35.3 How to run the compliance or preflight demo

Before version 10.1, Foxit PDF SDK provides a **compliance** demo located in the "`\examples\simple_demo\compliance`" folder to show you how to use Foxit PDF SDK to verify whether a PDF is compliance with PDF/A standard, and convert a PDF to be compliance with PDF/A standard, as well as convert PDF versions.

From version 10.1, Foxit PDF SDK provides two demos:

- A **compliance** demo located in the "`\examples\simple_demo\compliance`" folder to show you how to use Foxit PDF SDK to verify whether a PDF is compliance with PDF/A or PDF/E or PDF/X standard, and convert a PDF to be compliance with PDF/A or PDF/E or PDF/X standard, as well as convert PDF versions.
- A **preflight** demo located in the "`\examples\simple_demo\preflight`" folder to show you how to use Foxit PDF SDK to get preflight keys and analyze or fixup PDF file.

3.35.3.1 Build a compliance resource directory

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

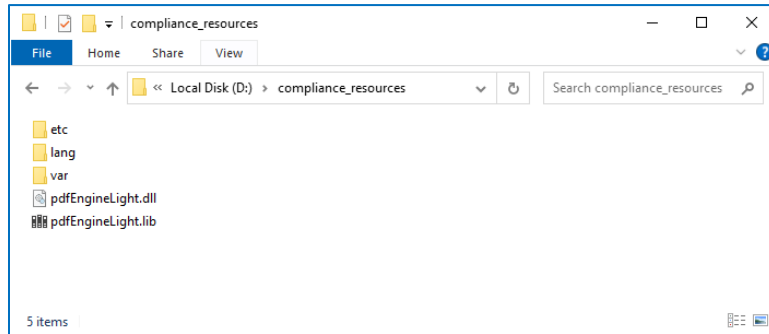
Starting from version 10.0, the compliance resource files provide default thread-safety. For multithreading, the API **FSDK.ComplianceEngine.InitializeThreadContext** should be called first for a new thread before using any other methods in the compliance add-on module.

Windows

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

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

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

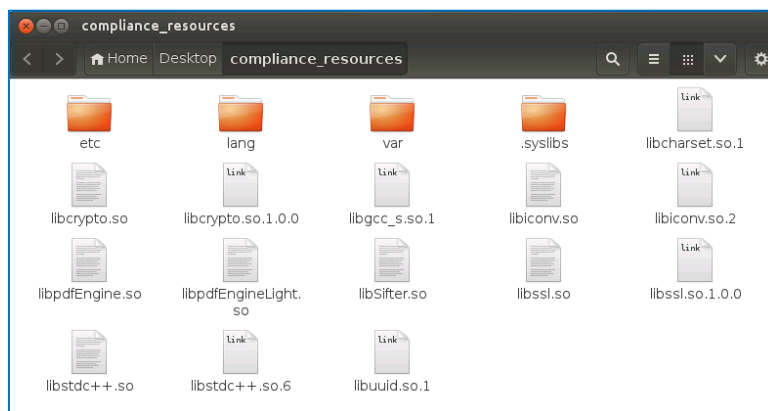


Linux (for Node.js, only support Linux x64)

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

- 1) Create a new folder to add the resources. For example, `"/root/Desktop/compliance_resources"`.
- 2) Copy the whole folders of **"ect"**, **"lang"**, **"var"** under the `"compliance/linux"` to `"/root/Desktop/compliance_resources"`.
- 3) Add the appropriate library resource based on the platform architecture.
 - For **linux32**, copy **all the files** under `"compliance/linux/lib/x86"` folder to `"/root/Desktop/compliance_resources"`.
 - For **linux64**, copy **all the files** under `"compliance/linux/lib/x64"` folder to `"/root/Desktop/compliance_resources"`.

For example, use **linux32** platform architecture, then the compliance resource directory should be as follows:



Note: For Linux platform, you should put the compliance resource directory into the search path for system shared library before running the demo, otherwise **FSDK.ComplianceEngine.Initialize** will fail.

For example, you can use the command (`export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/root/Desktop/compliance_resources`) to temporarily add the compliance resource directory to `LD_LIBRARY_PATH`.

3.35.3.2 Configure the demo

After building the compliance resource directory, configure the demo in the "`examples\simple_demo\compliance\compliance.js`" for compliance demo or "`examples\simple_demo\preflight\preflight.js`" for preflight demo.

This section takes **Windows** as an example to show you how to configure the demo in the "compliance.js" or "preflight.js" file. For Linux platform, do the same configuration with Windows.

Specify the compliance resource directory

In the "compliance.js" or "preflight.js" file, add the compliance resource directory as follows, which will be used to initialize the compliance engine.

```
// 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.
let compliance_resource_folder_path = "D:/compliance_resources"

let compliance_engine_unlockcode = "";
// Initialize compliance engine.
let error_code = FSDK.ComplianceEngine.Initialize(compliance_resource_folder_path,
compliance_engine_unlockcode);
```

Note:

- If you are using a trial key for Foxit PDF SDK, you do not need to authorize the compliance engine library.
- If you are using an authorization key for Foxit PDF SDK, Foxit sales team will send you an extra unlock code for initializing compliance engine library. Pass the unlock code to the **initialize** function "`FSDK.ComplianceEngine.Initialize(compliance_resource_folder_path, compliance_engine_unlockcode)`".

(Optional) Set language for compliance engine (only for compliance demo)

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

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

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

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

(Optional) Set a temp folder for compliance engine (only for compliance demo)

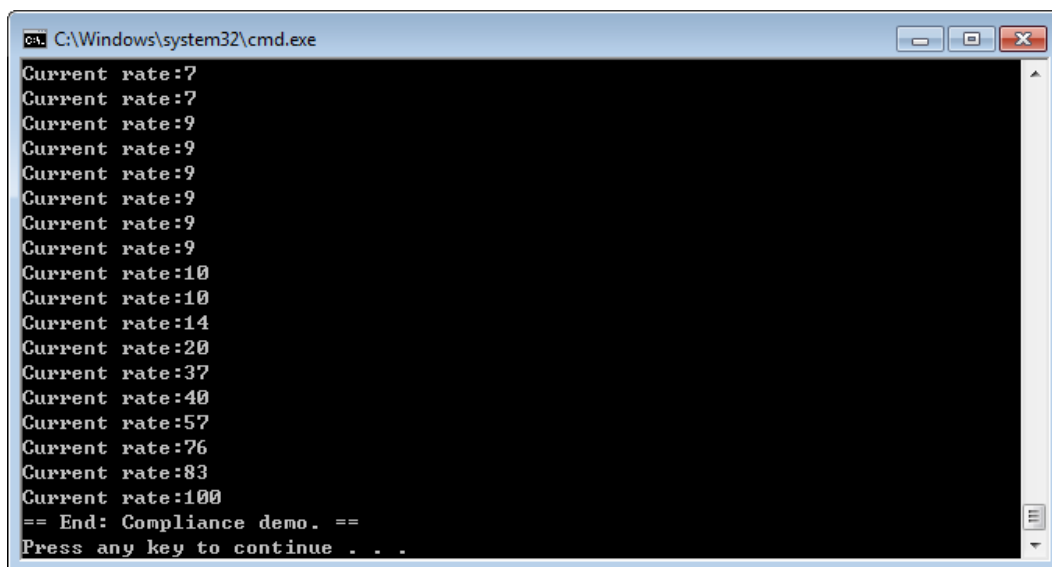
FSDK.ComplianceEngine.SetTempFolderPath function is used to set a temp folder to store several files for proper processing (e.g verifying or converting). If no custom temp folder is set by this function, the default temp folder in system will be used.

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

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

3.35.3.3 Run the demo

Take **compliance** demo as an example, once you run the demo successfully by "node compliance.js" in the CMD, the console will print the following by default:



```
C:\Windows\system32\cmd.exe  
Current rate:7  
Current rate:7  
Current rate:9  
Current rate:9  
Current rate:9  
Current rate:9  
Current rate:9  
Current rate:9  
Current rate:10  
Current rate:10  
Current rate:14  
Current rate:20  
Current rate:37  
Current rate:40  
Current rate:57  
Current rate:76  
Current rate:83  
Current rate:100  
== End: Compliance demo. ==  
Press any key to continue . . .
```

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

3.36 Optimization

Optimization feature can reduce the size of PDF files to save disk space and make files easier to send and store, through compressing images, deleting redundant data, discarding useless user data

and so on. Optimization module also provides functions to compress the color/grayscale/monochrome images in PDF files to reduce the size of the PDF files.

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

Example:

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

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let doc = new FSDK.PDFDoc(input_file);
let error_code = doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    console.log("The Doc [%s] Error: %d\n", input_path, error_code);
    return;
}
let pause = new Optimization_Pause(0, true);
let settings = new FSDK.OptimizerSettings();
let pause_callback = new FSDK.PauseCallback(pause);
settings.SetOptimizerOptions(FSDK.OptimizerSettings.e_OptimizerCompressImages);
let progressive = FSDK.Optimizer.Optimize(doc, settings, pause_callback);
let progress_state = FSDK.Progressive.e_ToBeContinued;
while (FSDK.Progressive.e_ToBeContinued === progress_state) {
    progress_state = progressive.Continue();
    let percent = progressive.GetRateOfProgress();
    let res_string = `Optimize progress percent:${percent} %`;
}
if (FSDK.Progressive.e_Finished === progress_state) {
    doc.SaveAs(output_directory + "ImageCompression_Optimized.pdf",
    FSDK.PDFDoc.e_SaveFlagRemoveRedundantObjects);
}
```

3.37 HTML to PDF Conversion

For some large HTML files or a webpage which contain(s) many contents, it is not convenient to print or archive them directly. Foxit PDF SDK provides APIs to convert the online webpage or local HTML files like invoices or reports into PDF file(s), which makes them easier to print or archive. In the process of conversion from HTML to PDF, Foxit PDF SDK supports to create and add PDF Tags based on the organizational structure of HTML. In addition, Foxit PDF SDK also supports to provide the generated files after HTML2PDF conversion in the form of file stream.

For HTML to PDF module, it supports HTML5, CSS3 and JavaScript.

Foxit PDF SDK supports to convert HTML to PDF on Windows and Linux (only for x86 and x64) platforms. But for HTML to PDF engine (Linux), the version of `libnss` should be 3.22.

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

3.37.1 System requirements

Platform: Windows, Linux (x86 and x64), Mac

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

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

SDK Version: Foxit PDF SDK (C++, Java, C#, Objective-C) 7.0 or higher; Foxit PDF SDK (C) 7.4 or higher; Foxit PDF SDK (Python) 8.3 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.37.2 HTML to PDF engine files

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

After getting the package, extract it to a desired directory (for example, extract the package to a directory: "**htmltopdf/win**" for Windows, and "**htmltopdf/linux**" for Linux

3.37.3 How to run the html2pdf demo

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

3.37.3.1 Prepare a HTML2PDF engine directory

Before running the html2pdf demo, you should first extract engine package to a desired directory (for example, extract the package to a directory: "**D:/htmltopdf/win/**" for Windows), and then pass the engine file path to the API **FSDK.Convert.FromHTML** to convert html to PDF file.

3.37.3.2 Configure the demo

For html2pdf demo, you can configure the demo in the "\\examples\\simple_demo\\html2pdf**html2pdf.js**" file, or you can configure the demo with parameters directly in a command prompt or a terminal. Following will configure the demo in "html2pdf.js" file on Windows for example. For Linux platform, do the same configuration with Windows.

Specify the html2pdf engine directory

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

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

(Optional) Specify cookies file path

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

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

3.37.3.3 Run the demo

Run the demo without parameters

Once you run the demo successfully by "node html2pdf.js" in the CMD, the console will print the following by default:

```
D:\foxitpdfsdk_..._win_nodejs_example\examples\simple_demo\html2pdf>node html2pdf.js  
Convert HTML to PDF successfully.  
D:\foxitpdfsdk_..._win_nodejs_example\examples\simple_demo\html2pdf>|
```

Run the demo with parameters

After building the demo successfully, open a command prompt, navigate to "`\examples\simple_demo\html2pdf`", type "`node html2pdf.js --help`" for example to see how to use the parameters to execute the program.

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

```
D:\foxitpdfsdk_..._win_nodejs_example\examples\simple_demo\html2pdf>node html2pdf.js -htm  
l www.foxit.com -w 900 -h 300  
Convert HTML to PDF successfully.  
D:\foxitpdfsdk_..._win_nodejs_example\examples\simple_demo\html2pdf>
```

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

Parameters Description

Basic Syntax:

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>]

Note:

- <> required
- [] optional

Parameters	Description
--help	The usage description of the parameters.
-html	The url or html file path. For examples '-html www.foxitsoftware.com'.
-o	The path of the output PDF file.
-engine	The path of the engine file "fxhtml2pdf.exe".
-w	The page width of the output PDF file in points.
-h	The page height of the output PDF file in points.
-r	The page rotation for the output PDF file. <ul style="list-style-type: none"> • 0 : 0 degree. • 1 : 90 degree. • 2 : 180 degree. • 3 : 270 degree.
-ml	The left margin of the pages for the output PDF file.
-mr	The right margin of the pages for the output PDF file.
-mt	The top margin of the pages for the output PDF file.
-mb	The bottom margin of the pages for the output PDF file.
-mode	The page mode for the output PDF file. <ul style="list-style-type: none"> • 0 : Single page mode.

Parameters	Description
	<ul style="list-style-type: none">• 1 : Multiple pages mode.
-scale	The scaling mode. <ul style="list-style-type: none">• 0 : No need to scale pages.• 1 : Scale pages.• 2 : Enlarge page.
-link	Whether to convert links. <ul style="list-style-type: none">• 'yes' : Convert links.• 'no' : No need to convert links.
-tag	Whether to generate tag. <ul style="list-style-type: none">• 'yes' : Generate tag.• 'no' : No need to generate tag.
-bookmarks	Whether to generate bookmarks. <ul style="list-style-type: none">• 'yes' : Generate bookmarks .• 'no' : No need to generate bookmarks.
-print_background	Whether to print background. <ul style="list-style-type: none">• 'yes' : Print bookmarks .• 'no' : No need to print bookmarks.
-optimize_tag	Whether to optimize tag tree. <ul style="list-style-type: none">• 'yes' : Optimize tag tree .• 'no' : No need to optimize tag tree.
-media	The media style. <ul style="list-style-type: none">• 0 : Screen media style.• 1 : Print media style.
-encoding	The HTML encoding format. <ul style="list-style-type: none">• 0 : Auto encoding .• 1-73 : Other encodings.
-render_images	Whether to render images. <ul style="list-style-type: none">• 'yes' : Render images.• 'no' : No need to render images.
-remove_underline_for_link	Whether to remove underline for link. <ul style="list-style-type: none">• 'yes' : Remove underline for link.• 'no' : No need to remove underline for link.
-headerfooter	Whether to generate headerfooter. <ul style="list-style-type: none">• 'yes' : Generate headerfooter.• 'no' : No need to generate headerfooter.

Parameters	Description
-headerfooter_title	The headerfooter title.
-headerfooter_url	The headerfooter url.
-bookmark_root_name	The bookmark root name.
-resize_objects	Whether to enable the JavaScripts related resizing of the objects during rendering process. <ul style="list-style-type: none"> 'yes' : Enable. 'no' : Disable.
-cookies	The path of the cookies file exported from a URL that you want to convert.
-timeout	The timeout of loading webpages.

3.37.4 How to work with Html2PDF API

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
let pdf_setting_data = new FSDK.HTML2PDFSettingData();
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 = FSDK.e_Rotation0;
pdf_setting_data.page_height = 640;
pdf_setting_data.page_width = 900;
pdf_setting_data.page_mode = FSDK.HTML2PDFSettingData.e_PageModeSinglePage;
pdf_setting_data.scaling_mode = FSDK.HTML2PDFSettingData.e_ScalingModeScale;
pdf_setting_data.to_print_background = true;
pdf_setting_data.to_optimize_tag_tree = false;
pdf_setting_data.media_style = FSDK.HTML2PDFSettingData.e_MediaStyleScreen;
...
FSDK.Convert.FromHTML(url_or_html, engine_path, cookies_path, pdf_setting_data, output_file_path, time_out);
```

3.37.5 How to get HTML data from stream and convert it to a PDF file

1. Defines a `FSDK.FileRead` class inherited from `FSDK.FileReaderCallback` used to get html data from stream or memory. And defines a `FSDK.FileWriter` class inherited from `FSDK.FileWriterCallback` used to do file writing. For the implementations of `FSDK.FileRead` and `FSDK.FileWriter` classes, please refer to the **html2pdf** demo in the "`examples\simple_demo\html2pdf`" folder.
2. Get html data from stream and set resources related to source html.
3. Call the `FSDK.Convert.FromHTML` function to convert it to a PDF file.

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

let pdf_setting_data = new FSDK.HTML2PDFSettingData();
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 = new FSDK.RectF(18, 18, 18, 18);
pdf_setting_data.is_convert_link = true;
pdf_setting_data.rotate_degrees = FSDK.e_Rotation0;
pdf_setting_data.is_generate_tag = true;

let mode = FSDK.HTML2PDFSettingData.e_PageModeSinglePage;
pdf_setting_data.to_generate_bookmarks = true;
let scale = FSDK.HTML2PDFSettingData.e_ScalingModeNone;
pdf_setting_data.encoding_format = FSDK.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 = "";
pdf_setting_data.headerfooter_url = "";
pdf_setting_data.bookmark_root_name = "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 = FSDK.HTML2PDFSettingData.e_MediaStyleScreen;
pdf_setting_data.to_load_active_content = false;
let output_path = output_directory + "html2pdf_filestream_result.pdf";
filewrite = new FileWriter();
let write_callback = new FSDK.FileWriterCallback(filewrite);
filewrite.LoadFile(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.
let htmlfile = "";
filereader = new FileRead();
filereader.LoadFile(htmlfile);
let read_callback = new FSDK.FileReaderCallback(filereader);

let html2PDFRelatedResourceArray = new FSDK.HTML2PDFRelatedResourceArray ;
let html2PDFRelatedResource = new FSDK.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.
let htmlfilepng = "";
let filereader1 = new FileRead();
filereader1.LoadFile(htmlfilepng);
let read_callback1 = new FSDK.FileReaderCallback(filereader1);
html2PDFRelatedResource.related_resource_file = read_callback1;
// "relativefilepath" is the resource file's relative path. For example: "./aaa.png".
let relativefilepath = "";
html2PDFRelatedResource.resource_file_relative_path = relativefilepath;
html2PDFRelatedResourceArray.Add(html2PDFRelatedResource);
```

```
FSDK.Convert.FromHTML(read_callback, html2PDFRelatedResourceArray, engine_path, null, pdf_setting_data, write_callback, 30);
```

3.38 Office to PDF Conversion with third-party engines

Foxit PDF SDK provides APIs to convert Microsoft Office documents (Word, Excel and PowerPoint) into professional-quality PDF files on Windows and Linux (x86, x64 and armv8) platforms.

Foxit PDF SDK for Node.js supports Windows and Linux x64 platforms.

For using this feature, please note that:

- Make sure that Microsoft Office 2007 version or higher is already installed on your Windows system.
- Before converting Excel to PDF, make sure that the default Microsoft virtual printer is already set on your Windows system.
- For Linux x86/x64, make sure that LibreOffice is already installed on your Linux system.

Note: When using LibreOffice 7.0 or a higher version, if you encounter an error like "An unknown error has occurred", you can try to set an environment variable before running the program as follows:

```
"export URE_BOOTSTRAP=vnd.sun.star.pathname:/opt/libreoffice7.x/program/fundamentalrc"
```

Where, 'x' represents the LibreOffice version.

3.38.1 System requirements

Platform: Windows, Linux (x86, x64 and armv8)

Programming Language: C, C++, Python, Java, C#, Node.js

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

SDK Version: Word and Excel (Foxit PDF SDK (C++, C#, Java) 7.3 or higher, Foxit PDF SDK (C) 7.4 or higher, Foxit PDF SDK (Python) 8.3 or higher), PowerPoint (Foxit PDF SDK (C, C++, C#, Java) 7.4 or higher, Foxit PDF SDK (Python) 8.3 or higher), Word/Excel/PowerPoint (Foxit PDF SDK (Node.js) 10.0 or higher)

Example:

Note: For Linux x86/x64, the parameter "`engine_path`" in the following sample code represents the path of LibreOffice engine. To get the installed path of LibreOffice, you can input the command "**locate soffice.bin**" in a terminal, then the path will be shown, for example,

"/usr/lib/libreoffice/program/soffice.bin". Then the value of "engine_path" parameter is set to "/usr/lib/libreoffice/program".

3.38.2 How to convert Word to PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let word_file_path = "test.doc";
let output_path = "saved.pdf";

// Use default Word2PDFSettingData values.
let word_convert_setting_data = new FSDK.Word2PDFSettingData();
if (process.platform === 'win32') {
  FSDK.Convert.FromWord(word_file_path, "", output_path, word_convert_setting_data);
} else {
  engine_path = ""; // Please fill in the correct engine path. For example, "/usr/lib/libreoffice/program".
  FSDK.Convert.FromWord(word_file_path, "", output_path, engine_path, word_convert_setting_data);
}
```

3.38.3 How to convert Excel to PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let excel_file_path = "test.xlsx";
let output_path = "saved.pdf";

// Use default Excel2PDFSettingData values.
let excel_convert_setting_data = new FSDK.Excel2PDFSettingData();
if (process.platform === 'win32') {
  FSDK.Convert.FromExcel(excel_file_path, "", output_path, excel_convert_setting_data);
} else {
  FSDK.Convert.FromExcel(excel_file_path, "", output_path, engine_path, excel_convert_setting_data);
}
```

3.38.4 How to convert PowerPoint to PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let ppt_file_path = "test.pptx";
let output_path = "saved.pdf";

// Use default PowerPoint2PDFSettingData values.
let ppt_convert_setting_data = new FSDK.PowerPoint2PDFSettingData();
```

```
if (process.platform === 'win32') {  
    FSDK.Convert.FromPowerPoint(ppt_file_path, "", output_path, ppt_convert_setting_data);  
} else {  
    FSDK.Convert.FromPowerPoint(ppt_file_path, "", output_path, engine_path, ppt_convert_setting_data);  
}
```

3.39 Office to PDF Conversion without third-party engines

From version 10.1, Foxit PDF SDK offers the capability to convert Microsoft Office documents (Word, Excel and PowerPoint) into professional-quality PDF files without any third-party engine. This feature is available through the Foxit PDF Conversion SDK on Windows platform.

3.39.1 System requirements

Platform: Windows

Programming Language: C, C++, Python, Java, C#, Node.js

License Key requirement: 'Office2PDF' module permission in the license key

SDK Version: Foxit PDF SDK 10.1

3.39.2 Office to PDF resource files (Foxit PDF Conversion SDK)

Please contact Foxit support team or sales team to get the Foxit PDF Conversion SDK (C++).

After getting Foxit PDF Conversion SDK package, extract it to a desired directory, for example, extract the package to a directory: "**D:/foxitpdfconversionsdk*_win/**" for Windows.

3.39.3 How to run the office2pdf demo using Foxit PDF Conversion SDK

Before running the office2pdf demo in the "\examples\simple_demo\office2pdf" folder using Foxit PDF Conversion SDK, you should first add the Foxit PDF Conversion SDK library in the demo code, for example:

```
// If you want to convert office files to PDF whitout other third-party engines, you can use the Office2PDF  
module.  
let library_path = "D:/foxitpdfconversionsdk*_win/lib/fpdfconversionsdk_win32.dll";
```

Then, specify the office2pdf resource data files:

```
// A valid path of a folder which contains resource data files.  
office2pdf_setting_data.resource_folder_path = "D:/foxitpdfconversionsdk*_win/res/office2pdf";
```

Finally, run the demo following the steps as the other demos.

3.39.4 How to convert office files to PDF without third-party engines

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

// If you want to convert office files to PDF without other third-party engines, you can use the Office2PDF
// module.
let library_path = ""; // Path of Foxit PDF Conversion SDK library, please ensure the path is valid.
// Initialize the Office2PDF module.
FSDK.Office2PDF.Initialize(library_path);
// Use default Office2PDFSettingData values.
let office2pdf_setting_data = new FSDK.Office2PDFSettingData();
// A valid path of a folder which contains resource data files.
office2pdf_setting_data.resource_folder_path = "";
// Convert Word file to PDF file.
output_path = output_directory + "word2pdf_result_foxit.pdf";
FSDK.Office2PDF.ConvertFromWord(word_file_path, "", output_path, office2pdf_setting_data);
// Convert Excel file to PDF file.
output_path = output_directory + "excel2pdf_result_foxit.pdf";
FSDK.Office2PDF.ConvertFromExcel(excel_file_path, "", output_path, office2pdf_setting_data);
// Convert PowerPoint file to PDF file.
output_path = output_directory + "ppt2pdf_result_foxit.pdf";
FSDK.Office2PDF.ConvertFromPowerPoint(ppt_file_path, "", output_path, office2pdf_setting_data);
// Release the Office2PDF module.
FSDK.Office2PDF.Release();
```

3.40 Output Preview

Foxit PDF SDK supports output preview feature which can preview color separations and test different color profiles.

Note: Currently, the output preview feature is not supported on the Linux ARM platform.

3.40.1 System requirements

Platform: Windows, Linux (x86 and x64), Mac (x64)

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

License Key requirement: valid license key

SDK Version: Foxit PDF SDK (C, C++, Java, C#, Objective-C) 7.4 or higher; Foxit PDF SDK (Python) 8.3 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.40.2 How to run the output preview demo

Before running the output preview demo in the "\examples\simple_demo\output_preview" folder, you should first set the folder path of "\res\icc_profile" in the SDK package to the variable **default_icc_folder_path**. For example:

```
// "default_icc_folder_path" is the path of the folder which contains default icc profile files. Please refer to  
Developer Guide for more details.  
let default_icc_folder_path = "E:/foxitpdfsdk_X_X_win_nodejs_example/res/icc_profile";
```

Then, run the demo following the steps as the other demos.

3.40.3 How to do output preview using Foxit PDF SDK

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
  
...  
// Make sure that SDK has already been initialized successfully.  
  
// Set folder path which contains default icc profile files.  
FSDK.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.  
let output_preview = new FSDK.OutputPreview(pdf_doc);  
let simulation_icc_file_path = input_path + "icc_profile/USWebCoatedSWOP.icc";  
output_preview.SetSimulationProfile(simulation_icc_file_path);  
output_preview.SetShowType(FSDK.OutputPreview.e_ShowAll);  
let process_plates = output_preview.GetPlates(FSDK.OutputPreview.e_ColorantTypeProcess);  
let spot_plates = output_preview.GetPlates(FSDK.OutputPreview.e_ColorantTypeSpot);  
// Set check status of spot plate to be true, if there's any spot plate.  
for (let i = 0; i < spot_plates.GetSize(); i++) {  
    output_preview.SetCheckStatus(spot_plates.GetAt(i), true);  
}  
  
// Generate preview bitmap  
// Only set one process plate to be checked each time and generate the preview bitmap.  
for (let i = 0; i < process_plates.GetSize(); i++) {  
    if (0 != i)  
        output_preview.SetCheckStatus(process_plates.GetAt(i-1), false);  
    output_preview.SetCheckStatus(process_plates.GetAt(i), true);  
    let preview_bitmap = output_preview.GeneratePreviewBitmap(pdf_page, display_matrix, renderer);  
}
```

3.41 Combination

Combination feature is used to combine several PDF files into one PDF file.

3.41.1 How to combine several PDF files into one PDF file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let file_info1 = new FSDK.CombineDocumentInfo (input_path + "Annot_all.pdf", "");
file_info1.SetBookmarkTitle("Annot_all.pdf");
file_info1.SetPDFFileName("Annot_all.pdf");

let pdfdoc = new FSDK.PDFDoc(input_path + "PDF2Img.pdf");
if (FSDK.e_ErrSuccess !== pdfdoc.Load("")) return 1;
let file_info2 = new FSDK.CombineDocumentInfo(pdfdoc);
file_info2.SetBookmarkTitle("PDF2Img.pdf");
file_info2.SetPDFFileName("PDF2Img.pdf");

let file_info3 = new FSDK.CombineDocumentInfo(input_path + "SamplePDF.pdf", "");
file_info3.SetBookmarkTitle("SamplePDF.pdf");
file_info3.SetPDFFileName("SamplePDF.pdf");

let combine_document_array = new FSDK.CombineDocumentInfoArray();
combine_document_array.Add(file_info1);
combine_document_array.Add(file_info2);
combine_document_array.Add(file_info3);

let option = FSDK.Combination.e_CombineDocsOptionBookmark |
FSDK.Combination.FSDK.e_CombineDocsOptionAcroformRename |
FSDK.Combination.e_CombineDocsOptionStructTree |
FSDK.Combination.e_CombineDocsOptionOutputIntents |
FSDK.Combination.e_CombineDocsOptionOCProperties | FSDK.Combination.e_CombineDocsOptionMarkInfos
|
FSDK.Combination.e_CombineDocsOptionPageLabels | FSDK.Combination.e_CombineDocsOptionNames |
FSDK.Combination.e_CombineDocsOptionObjectStream |
FSDK.Combination.e_CombineDocsOptionDuplicateStream;

FSDK.Combination.StartCombineDocuments(output_directory + "pdfcombination.pdf",
combine_document_array, option, null);
```

3.42 PDF Portfolio

PDF portfolios are a combination of files with different formats. Portfolio file itself is a PDF document, and files with different formats can be embedded into this kind of PDF document.

3.42.1 System requirements

Platform: Windows, Linux, Mac

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

License Key requirement: valid license key

SDK Version: Foxit PDF SDK (C, C++, Java, C#, Objective-C) 7.6 or higher; Foxit PDF SDK (Python) 8.3 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

Example:

3.42.2 How to create a new and blank PDF portfolio

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let new_portfolio = FSDK.Portfolio.CreatePortfolio();

// Set properties, add file/folder node to the new portfolio.
...

// Get portfolio PDF document object.
let portfolio_pdf_doc = new_portfolio.GetPortfolioPDFDoc();
```

3.42.3 How to create a Portfolio object from a PDF portfolio

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

let pdf_doc = new FSDK.PDFDoc(portfolio_file_path);
let error_code = pdf_doc.Load("");
if (error_code !== FSDK.e_ErrSuccess) {
    return;
}
if (false == pdf_doc.IsPortfolio()) {
    let exist_portfolio = FSDK.Portfolio.CreatePortfolio(pdf_doc);
}
```

3.42.4 How to get portfolio nodes

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// Make sure that SDK has already been initialized successfully.

// Portfolio object has been created, assume it is named "portfolio".
...

let root_node = exist_portfolio.GetRootNode();
let root_folder = new FSDK.PortfolioFolderNode(root_node);
let sub_nodes = root_folder.GetSortedSubNodes();
for (let index = 0; index < sub_nodes.GetSize(); index++) {
    let node = sub_nodes.GetAt(index);
    let node_type = new FSDK.PortfolioNode(node).GetNodeType();
}
```

```
switch (node_type) {
  case FSDK.PortfolioNode.e_TypeFolder:
    let folder_node = new FSDK.PortfolioFolderNode(node);
    break;
  case FSDK.PortfolioNode.e_TypeFile:
    let file_node = new FSDK.PortfolioFileNode(node);
    let file_spec = file_node.GetFileSpec();
    break;
  default:
    break;
}
```

3.42.5 How to add file node or folder node

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

...
// 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 a non-PDF file to root folder node.
let input_file_path = input_path + "FoxitLogo.jpg";
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
let new_sub_filenode = root_folder.AddFile(input_file_path);

// 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.
let custom_streamcallback = new FileStream();
let stream_callback= new FSDK.StreamCallback(custom_streamcallback);
let new_file_node_2 = root_folder.AddFile(stream_callback, "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".
...
let new_filenode_3 = new_sub_foldernode.AddPDFDoc(test_pdf_doc, "pdf_file_name");

// User can update properties of file specification for new_file_node_3 if necessary.
...

// Add a sub folder in root_folder.
let new_sub_foldernode = root_folder.AddSubFolder("Sub Folder-1");

// User can add file or folder node to new_sub_foldernode.
```

...

3.42.6 How to remove a node

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
// 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

Foxit PDF SDK supports to add table to PDF files.

3.43.1 System requirements

Platform: Windows, Mac, Linux

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

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

SDK Version: Foxit PDF SDK (C, C++, Java, C#, Python, Objective-C) 8.4 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.43.2 How to add table to a PDF document

Foxit PDF SDK provides an electronictable demo located in the "examples\simple_demo\electronictable" folder to show you how to use Foxit PDF SDK to add table to PDF document.

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...
let index = 0;
let cell_array = new FSDK.TableCellDataArray();
for (let row = 0; row < 4; row++) {
    let col_array = new FSDK.TableCellDataColArray();
    for (let col = 0; col < 3; col++) {
        style = GetTableTextStyle(index);
        let cell_text = GetTableCellText(index++);
        let cell_data = new FSDK.TableCellData(style, 0xFFFFFFFF, cell_text, new FSDK.Image(), new FSDK.RectF());
        col_array.Add(cell_data);
    }
    cell_array.Add(col_array);
}
```

```

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

let outside_border_left = new FSDK.TableBorderInfo();
outside_border_left.line_width = 1;
let outside_border_right = new FSDK.TableBorderInfo();
outside_border_right.line_width = 1;
let outside_border_top = new FSDK.TableBorderInfo();
outside_border_top.line_width = 1;
let outside_border_bottom = new FSDK.TableBorderInfo();
outside_border_bottom.line_width = 1;
let inside_border_row = new FSDK.TableBorderInfo();
inside_border_row.line_width = 1;
let inside_border_col = new FSDK.TableBorderInfo();
inside_border_col.line_width = 1;
let data = new FSDK.TableData(new FSDK.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, new FSDK.TableCellIndexArray(), new FSDK.FloatArray(), new FSDK.FloatArray());
FSDK.TableGtor.AddTableToPage(pageenera, data, cell_array);
...

```

3.44 Accessibility

Foxit PDF SDK supports to tag PDF files.

3.44.1 System requirements

Platform: Windows, Mac, Linux

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

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

SDK Version: Foxit PDF SDK (C, C++, Java, C#, Python, Objective-C) 8.4 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.44.2 How to tag a PDF document

Foxit PDF SDK provides a taggedpdf demo located in the "\\examples\\simple_demo\\taggedpdf" folder to show you how to use Foxit PDF SDK to tag a PDF document.

```

const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

let pdfDoc = new FSDK.PDFDoc(input_file);
pdfDoc.Load("");
let taggedpdf = new FSDK.TaggedPDF(pdfDoc);
let progressive = taggedpdf.StartTagDocument(null);
let progressState = FSDK.Progressive.e_ToBeContinued;
while (FSDK.Progressive.e_ToBeContinued == progressState)
    progressState = progressive.Continue();
pdfDoc.SaveAs(output_file_path, FSDK.PDFDoc.e_SaveFlagNormal);

```

3.45 PDF to Office Conversion

Foxit PDF SDK provides APIs to convert PDF files to MS office suite formats while maintaining the layout and format of your original documents on Windows and Linux platforms.

3.45.1 System requirements

Platform: Windows, Linux

Programming Language: C, C++, Java, Python, C#, Node.js

License Key requirement: 'PDF2Office' module permission in the license key

SDK Version: Foxit PDF SDK for Windows (C, C++, Java, Python, C#) 9.0 or higher; Foxit PDF SDK for Linux (C, C++, Java, Python, C#) 9.1 or higher; Foxit PDF SDK (Node.js) 10.0 or higher

3.45.2 PDF to Office resource files

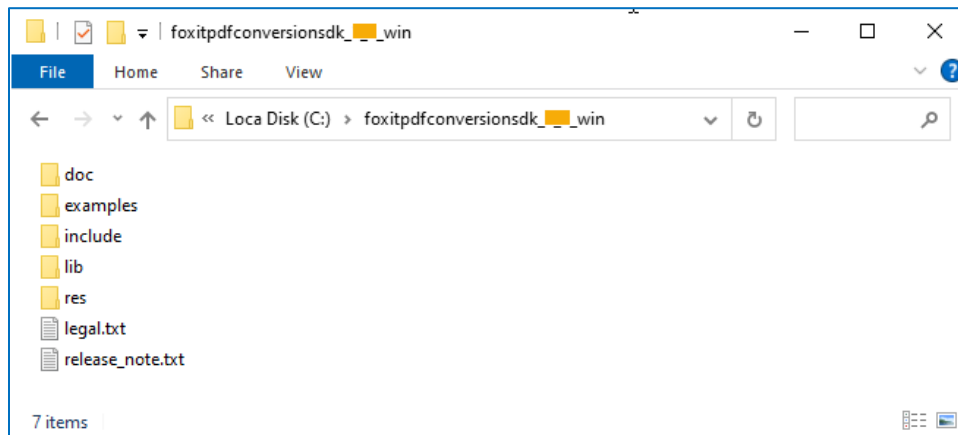
Please contact Foxit support team or sales team to get the PDF to Office resource files package naming Foxit PDF Conversion SDK (C++).

Note:

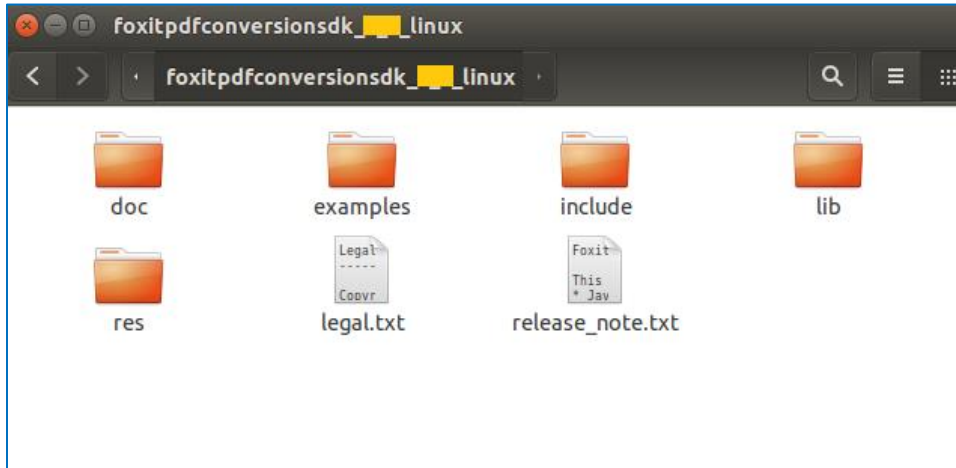
- From version 9.2, it requires Foxit PDF Conversion SDK 1.5 or higher.
- For version 9.0/9.1, it requires Foxit PDF Conversion SDK 1.4 or lower.

After getting Foxit PDF Conversion SDK package, extract it to a desired directory (for example, extract the package to a directory: `"/foxitpdfconversionsdk_*_win/"` for Windows, and `"/foxitpdfconversionsdk_*_Linux/"` for Linux x86/x64), and then you can see the resource files for PDF to Office are as follows:

For **Windows**:



For **Linux x86/x64**:



3.45.3 How to run the pdf2office demo

Foxit PDF SDK provides a pdf2office demo located in the "\examples\simple_demo\pdf2office" folder to show you how to use Foxit PDF SDK to convert PDF files to MS office suite formats.

3.45.3.1 Prepare a PDF2Office resource directory

Before running the pdf2office demo, you should first extract the PDF to Office resource files (Foxit PDF Conversion SDK) package to a desired directory (for example, extract the package to a directory: "**C:/foxitpdfconversionsdk_*_win/**" for Windows), and then pass the engine file located in "lib" folder to the API **FSDK.PDF2Office.Initialize** to initialize PDF2Office engine.

3.45.3.2 Configure the demo

For pdf2office demo, you can configure the demo in the "\examples\simple_demo\pdf2office\pdf2office.js" file. Following will configure the demo in "pdf2office.js" file on Windows for example. For Linux platform, do the similar configuration with Windows.

Specify the pdf2office engine directory

In the "pdf2office.js" file, add the path of the engine file "pdf2office" as follows, which will be used to convert PDF files to office files.

```
// Please ensure the path is valid.  
FSDK.PDF2Office.Initialize("C:/foxitpdfconversionsdk_*_win/lib/fpdfconversionsdk_win32.dll");
```

(Optional) Specify whether to enable machine learning-based recognition functionality

```
setting_data.enable_ml_recognition = false;
```

(Optional) Specify the page range to be converted

```
setting_data.page_range = new FSDK.Range();
```

(Optional) Specify whether to convert the comments in the PDF documents

```
setting_data.include_pdf_comments = true;
```

(Optional) Specify whether to retain the page layout for PDF to Word conversion

```
setting_data.word_setting_data.enable_retain_page_layout = false;
```

Note: Starting from version 10.1, the PDF to Office Conversion engine supports a timeout parameter. This parameter defines the maximum time allowed for the conversion process to complete. If the conversion exceeds the specified time, it will be terminated. The timeout must be a non-negative value. A value of 0 disables the timeout, allowing the conversion to proceed without any time limitation.

3.45.3.3 Run the demo

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

```
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.
```

The output files are located in "\\examples\\simple_demo\\output_files\\pdf2office" folder.

3.45.4 How to work with PDF2office API

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

class CustomConvertCallback {
  constructor() {

  }

  NeedToPause() {
    return true;
  }

  ProgressNotify(converted_count, total_count) {
  }
};

let custom_callback = new CustomConvertCallback();
let convert_callback = new FSDK.ConvertCallback(custom_callback);
```

```
let progressive = FSDK.PDF2Office.StartConvertToWord(input_path + "word.pdf", "", output_directory +
"pdf2word_result.docx", setting_data, convert_callback);
if (progressive.GetRateOfProgress() != 100) {
    let state = FSDK.Progressive.e_ToBeContinued;
    while (FSDK.Progressive.e_ToBeContinued == state) {
        state = progressive.Continue();
    }
}

progressive = FSDK.PDF2Office.StartConvertToWord(reader_callback_word, "", stream_callback_word,
setting_data, convert_callback_word_stream);
if (progressive.GetRateOfProgress() != 100) {
    let state = FSDK.Progressive.e_ToBeContinued;
    while (FSDK.Progressive.e_ToBeContinued == state) {
        state = progressive.Continue();
    }
}

progressive = FSDK.PDF2Office.StartConvertToExcel(reader_callback_excel, "", stream_callback_excel,
setting_data, convert_callback_excel_stream);
if (progressive.GetRateOfProgress() != 100) {
    let state = FSDK.Progressive.e_ToBeContinued;
    while (FSDK.Progressive.e_ToBeContinued == state) {
        state = progressive.Continue();
    }
}
```

3.46 DWG to PDF Conversion

Foxit PDF SDK supports to convert DWG files to PDF files. If you want to use this feature, you should contact Foxit support team or sales team to get the engine files package.

3.46.1 System requirements

Platform: Windows, Linux (x86 and x64), Mac(x64)

Programming Language: C, C++, Java, C#, Python, Objective-C, Node.js

License Key requirement: 'DWG2PDF' module permission in the license key

SDK Version: Foxit PDF SDK 10.0 or higher

3.46.2 DWG To PDF engine files

Please contact Foxit support team or sales team to get the DWG to PDF engine files package. After getting the package, extract it to the desired directory. For example, extract the package to a directory: "**D:/dwgtopdf/win**" for Windows, and "**dwgtopdf/linux**" for Linux.

3.46.3 How to run the dwg2pdf demo

Before running the dwg2pdf demo in the "\examples\simple_demo\dwg2pdf" folder, you should first add the dwg2pdf engine file in the demo code, for example:

```
// "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.  
let engine_path = "D:/dwgtopdf/win";
```

Note: For Linux (x86 and x64), before running the demo, you should add the path of the dwg2pdf engine file to `LD_LIBRARY_PATH` environment variable.

```
export LD_LIBRARY_PATH=/dwgtopdf/linux:$LD_LIBRARY_PATH
```

Then, run the demo following the steps as the other demos.

3.46.4 How to convert DWG to PDF

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
  
let settings = new FSDK.DWG2PDFSettingData();  
settings.export_flags = FSDK.DWG2PDFSettingData.e_FlagEmbeddedTTF;  
settings.export_hatches_type = FSDK.DWG2PDFSettingData.e_DWG2PDFExportHatchesTypeBitmap;  
settings.other_export_hatches_type = FSDK.DWG2PDFSettingData.e_DWG2PDFExportHatchesTypeBitmap;  
settings.gradient_export_hatches_type = FSDK.DWG2PDFSettingData.e_DWG2PDFExportHatchesTypeBitmap;  
settings.searchable_text_type = FSDK.DWG2PDFSettingData.e_DWG2PDFSearchableTextTypeNoSearch;  
settings.is_active_layout = false;  
settings.paper_width = 640;  
settings.paper_height = 900;  
FSDK.Convert.FromDWG(engine_path, dwg_file_path, output_file_path, settings);
```

3.47 OFD

OFD files, standing for Open Financial Document, are used for storing and exchanging digital financial documents. They are open and XML-based, making them specifically designed for financial documents like contracts, invoices, and statements.

OFD files contain structured data and graphical elements defining the document's layout and content, including text, images, vector graphics, annotations, and other related information. The XML format facilitates easy interpretation, manipulation, and rendering of the document's content.

OFD files provide various benefits, including document integrity, security, and interoperability. They can be digitally signed to ensure authenticity and can be encrypted to protect sensitive information. OFD files also support interactive features like form fields and digital signatures.

To work with OFD files, OFD viewer or editor software that supports the OFD standard is needed. These tools allow you to display, edit, convert, and print the contents of OFD documents.

In essence, OFD files provide a standardized and efficient method for representing financial documents digitally, simplifying the exchange, storage, and management of financial information.

3.47.1 System requirements

Platform: Windows, Linux (x64 and armv8)

Programming Language: C, C++, Java, C#, Python, Node.js

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

SDK Version: Foxit PDF SDK 10.0 or higher

3.47.2 OFD engine file

Please contact Foxit support team or sales team to get the OFD engine file package.

After getting the package, extract it to the desired directory. For example, extract the package to a directory: "**D:/ofd/win**" for Windows, and "**ofd/linux64**" for Linux x64.

3.47.3 How to run the ofd demo

Before running the ofd demo in the "\examples\simple_demo\ofd" folder, you should first add the ofd engine file in the demo code, for example:

```
// "ofd_resource_path" is the path of ocr resources. Please refer to Developer Guide for more details.  
let ofd_resource_path = "D:/ofd/win/x64"; // For Windows x64
```

Then, run the demo following the steps as the other demos.

3.47.4 How to implement the conversion between OFD file and PDF file

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");  
  
// Initialize OFD engine.  
FSDK.Library.InitializeOFDEngine(ofd_resource_path);  
let src_ofd_path = input_path + "wm_txttiled.ofd";  
let src_pdf_path = input_path + "test.pdf";  
// Convert PDF document to OFD document, and convert OFD document to PDF document.  
let convert_param = new FSDK.OFDConvertParam(false);  
// Convert OFD document to PDF document.  
FSDK.Convert.FromOFD(src_ofd_path, "", output_directory + "ofd2pdf.pdf", convert_param);  
// Convert PDF document to OFD document.  
FSDK.Convert.ToOFD(src_pdf_path, "", output_directory + "pdf2ofd.ofd", convert_param);  
  
// Release OFD engine.  
FSDK.Library.ReleaseOFDEngine();
```

3.47.5 How to render OFD page

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");

// Initialize OFD engine.
FSDK.Library.InitializeOFDEngine(ofd_resource_path);
// Render OFD document to bitmap.
let render_file_path = input_path + "content_flag.ofd";
let doc = new FSDK.OFDDoc(render_file_path, "");
let ofd_page = doc.GetPage(0);
// Get the size of the page.
let w = ofd_page.GetWidth();
let h = ofd_page.GetHeight();
let bitmap = new FSDK.Bitmap(w, h, FSDK.Bitmap.e_DIBArgb, null, 0);
let fRect = new FSDK.Rect(0, 0, w, h);
bitmap.FillRect(0xFFFFFFFF, fRect);
// Get the display matrix of the page.
let matrix_1 = ofd_page.GetDisplayMatrix(0, 0, w, h, FSDK.e_Rotation0);

let ofd_render = new FSDK.OFDRenderer(bitmap);

let progressive = ofd_render.StartRender(ofd_page, matrix_1);
let sSaveFilePath = output_directory + "renderBitmap.bmp";
// Add the bitmap to image and save the image.
let image = new FSDK.Image();
image.AddFrame(bitmap);
image.SaveAs(sSaveFilePath);
ofd_page.Release();
doc.Release();
```

FAQ

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

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

- 1) Open a PDF file.
- 2) Load PDF pages and get the page objects.
- 3) Use **FSDK.PDFPage.GetGraphicsObjectAtPoint** to get the text object at a certain position. Note: use the page object to get rectangle to see the position of the text object.
- 4) Change the contents of the text objects and save the PDF document.

Following is the sample code:

```
const FSDK = require("@foxitsoftware/foxit-pdf-sdk-node");
...

function ChangeTextObjectContent() {
  let input_file = input_path + "AboutFoxit.pdf";
  let doc = new FSDK.PDFDoc(input_file);
  let error_code = doc.Load("");
  if (error_code != FSDK.e_ErrSuccess) {
    return false;
  }
  // Get original shading objects from the first PDF page.
  let original_page = doc.GetPage(0);
  original_page.StartParse(FSDK.e_ParsePageNormal, null, false);
  let pointf = new FSDK.PointF(92, 762);
  let arr = original_page.GetGraphicsObjectsAtPoint(pointf, 10, e_TypeText);
  for(let i = 0; i<arr.GetSize(); i++) {
    let graphobj = arr.GetAt(i);
    let textobj = graphobj.GetTextObject();
    textobj.SetText("Foxit Test");
  }
  original_page.GenerateContent();

  let output_directory = output_path + "graphics_objects/";
  let output_file = output_directory + "After_revise.pdf";
  doc.SaveAs(output_file, e_SaveFlagNormal);
  return true;
}
...
```

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

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

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

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

3. Why do I encounter "Fail to initialize the engine file or cannot load the engine file" for OCR and DWG2PDF modules on Windows 7 when running the corresponding simple demos, even if the engine files have been upgraded to the latest and the simple demos have configured the engine path correctly?

For Windows 7, you need to copy the dll files starting with **api-ms-win*** and the **ucrtbase.dll** in the engine directory to the system directory.

- If you are using a 32-bit engine and running on a 32-bit system, you need to copy the api-ms-win*.dll files and ucrtbase.dll from the engine directory to "C:/Windows/System32".
- If you are using a 32-bit engine and running on a 64-bit system, you need to copy the api-ms-win*.dll files and ucrtbase.dll from the engine directory to "C:/Windows/SysWOW64".
- If you are using a 64-bit engine, you need to copy the api-ms-win*.dll files and ucrtbase.dll from the engine directory to "C:/Windows/System32".

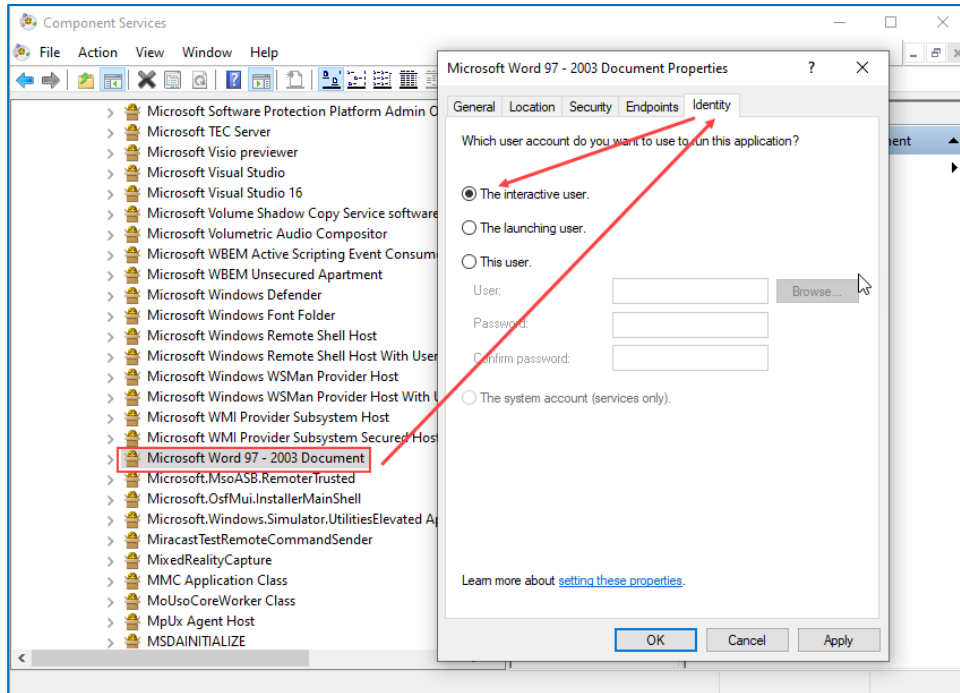
4. How to run the Office2PDF functionality in Windows services?

To run the Office2PDF functionality in Windows services, you need to configure the Office Component Services and permissions.

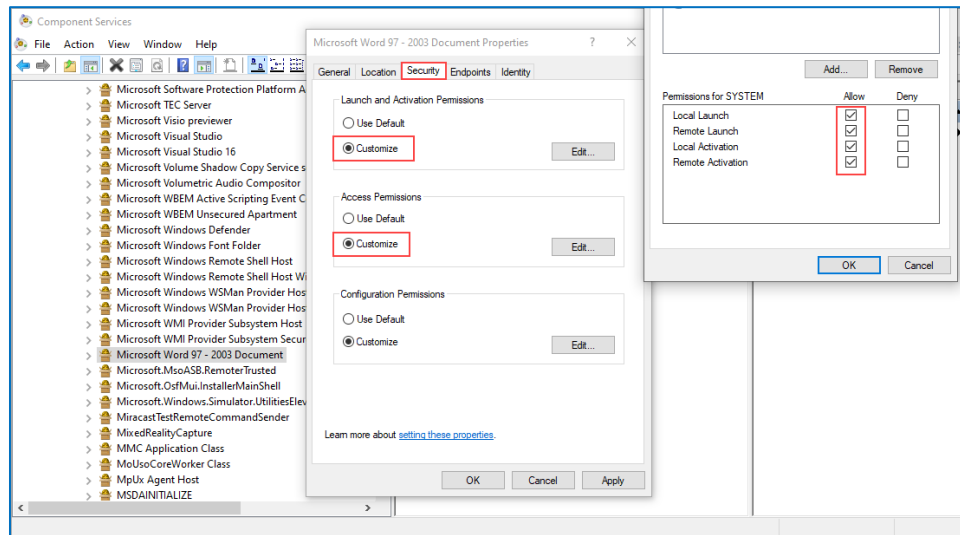
Take the Word component as an example.

- 1) Press **Win+R**, and then type **Dcomcnfg** to open Component Services, find [Component Services] -> [Computers] -> [My Computer] -> [DCOM Config] -> [Microsoft Word 97-2003 Document], right-click it and select Properties. Choose [Identity], and set it to "**The interactive user**".

Note: if you can't find [Microsoft Word 97-2003 Document] with **Dcomcnfg** command, you can try to use the "**comexp.msc -32**" command.



- 2) Set permissions. click [Security], set the [Launch and Activation Permissions] and [Access Permissions] to **Customize**. Click **Edit**, and add the current login account of the system and enable all permissions.



- 3) After finishing the above settings, the Word2PDF functionality can be run in the Windows services.

APPENDIX

Supported JavaScript List

Objects' property or method

Object	Properties/Method Names	Minimum Supported SDK Version
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

Object	Properties/Method Names	Minimum Supported SDK Version
	leaderLength	V9.0
	lineEnding	V9.0
	lock	V9.0
	notelcon	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

Object	Properties/Method Names	Minimum Supported SDK Version
app methods	alert	V4.0
	beep	V4.0
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	baseUrl	V4.0
	bookmarkRoot	V7.0
	calculate	V4.0
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	calculateNow	V4.0
	deletePages	V4.0
	exportAsFDF	V4.0
	flattenPages	V7.1
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	removeLinks	V8.4
	getLinks	V8.4
	importIcon	V8.4
	removeIcon	V8.4
	addWeblinks	V8.4
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	shift	V4.0
	source	V4.0
	target	V4.0
	targetName	V4.0
	type	V4.0
	value	V4.0
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	password	V4.0
	print	V4.0
	radiosInUnison	V4.0
	readonly	V4.0
	rect	V4.0
	required	V4.0
	richText	V4.0
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	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
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

Object	Properties/Method Names	Minimum Supported SDK Version
	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
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	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
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	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
MediaSettings properties	autoPlay	V8.4
	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

Object	Properties/Method Names	Minimum Supported SDK Version
	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
	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

Global methods

Method Names	Minimum Supported SDK Version
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

Method Names	Minimum Supported SDK Version
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
AFParseDateEx	V4.0
AFExtractNums	V4.0

REFERENCES

[1] PDF reference 1.7

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502

[2] PDF reference 2.0

<https://www.iso.org/standard/63534.html>

[3] Foxit PDF SDK API reference

sdk_folder/doc/Foxit PDF SDK Node.js API Reference.html

Note: sdk_folder is the directory of unzipped package.

SUPPORT

Foxit Support

In order to provide you with a more personalized support for a resolution, please log in to your [Foxit account](#) and submit a ticket so that we can collect details about your issue. We will work to get your problem solved as quickly as we can once your ticket is routed to our support team.

You can also check out our [Support Center](#), choose Foxit PDF SDK which also has a lot of helpful articles that may help with solving your issue.

Phone Support:

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