# 1. Overview

ComPDFKit Conversion SDK is a high-performance library designed for extracting and transforming the data within your PDF files, such as text, images, tables, links, and annotations, into various file formats. Our Conversion SDK retains the original document layout and the properties of the file data, ensuring a seamless document conversion experience.

Effortlessly integrate the ComPDFKit Conversion SDK into your projects in just a few steps, and enable the following file format conversions:

- Convert PDF to Word (.docx)
- Convert PDF to Excel (.xlsx)
- Convert PDF to PowerPoint (.pptx)
- Convert PDF to HTML (.html)
- Convert PDF to CSV (.csv)
- Convert PDF to Image (.png, .jpg, .jpeg, .bmp, .tiff)
- Convert PDF to Plain Text (.txt)
- Convert PDF to Rich Text Format (.rtf)
- Convert PDF to Structured Data (.json)
- Convert PDF to Markdown (.md)

Harness the power of ComPDFKit Conversion SDK and take your document processing to the next level with intelligent tools designed for accuracy and efficiency. To enhance your format conversion results, ComPDFKit offers AI-powered document tools with the following capabilities:

- Optical Character Recognition (OCR)
- Layout Analysis
- Table Recognition

## 1.1 Why ComPDFKit Conversion SDK

- Mature Technology

  With years of technology accumulation, we have established a complete mechanism of product iteration to offer a continuous guarantee for product competitiveness.

- Complete PDF and Format Conversion Functionalities

Our comprehensive features can meet diverse needs and are easy for our customers to use without training costs.

- High-quality Service

  professional service and technical support to quickly respond to users' feedback through onsite service or remote support like telephone, email, etc.

- Independent Intellectual

  Property Rights
  Our technology is independent and compliant with ISO, helping enterprises conduct international business without considering copyright risks.

### 1.2 ComPDFKit Conversion SDK

The ComPDFKit Conversion SDK is designed to facilitate the effortless conversion of PDF files into various other formats, all while preserving the original layout and formatting of the documents. In this guide, we will explore the ComPDFKit Conversion SDK and demonstrate how to utilize it within your Windows projects.

### 1.3 License & Trial

The ComPDFKit Conversion SDK is a commercial SDK that requires a license to grant developers the right to develop and distribute their applications. In development mode, each license is only valid for one device ID. ComPDFKit provides flexible licensing models, please contact our marketing team for more information. However, even if you have a license, it is prohibited to distribute any documents, sample code, or source code of the ComPDFKit Conversion SDK to any third parties.

If you do not have a License, please feel free to contact the ComPDFKit Team to obtain a License to try the ComPDFKit Conversion SDK.

## 2. Getting Started

With just a few lines of code, you can easily integrate the ComPDFKit Conversion SDK into your project. In this section, we will introduce the package structure of the ComPDFKit Conversion SDK for Java, system requirements for running it, methods of integration, as well as how to run the Demo.

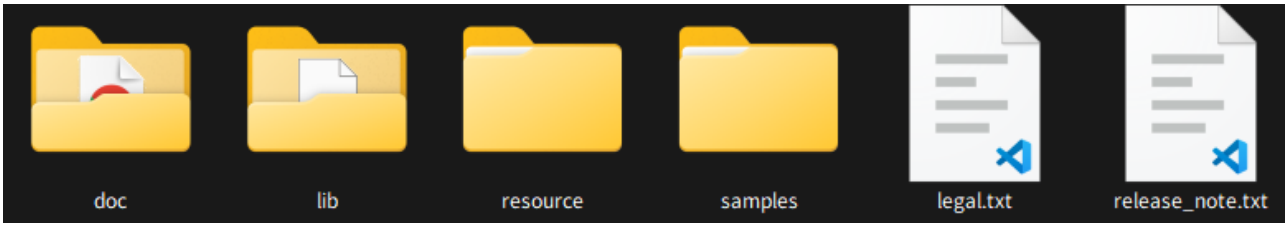
## 2.1 System Requirements

| DEVELOPMENT PLATFORM | SYSTEM REQUIREMENTS | DEVELOPMENT ENVIRONMENT | NOTICE |
| --- | --- | --- | --- |
| Android | - Android 7.0(API Level 24)or higher | Android Studio | - Only support on arm64-v8a and armeabi-v7a |

## 2.2 SDK Package Structure

The ComPDFKit Conversion SDK for Java SDK contains the following files:

- ***"doc"*** - API reference and developer guide.
- ***"lib"*** - Contains ComPDFKit Conversion Java SDK JAR.
- ***"samples"*** - A folder containing sample projects.
- ***"resource"*** - Contains font files and OCR models.
- ***"release_notes.txt"*** - Release information.
- ***"legal.txt"*** - Legal and copyright information.



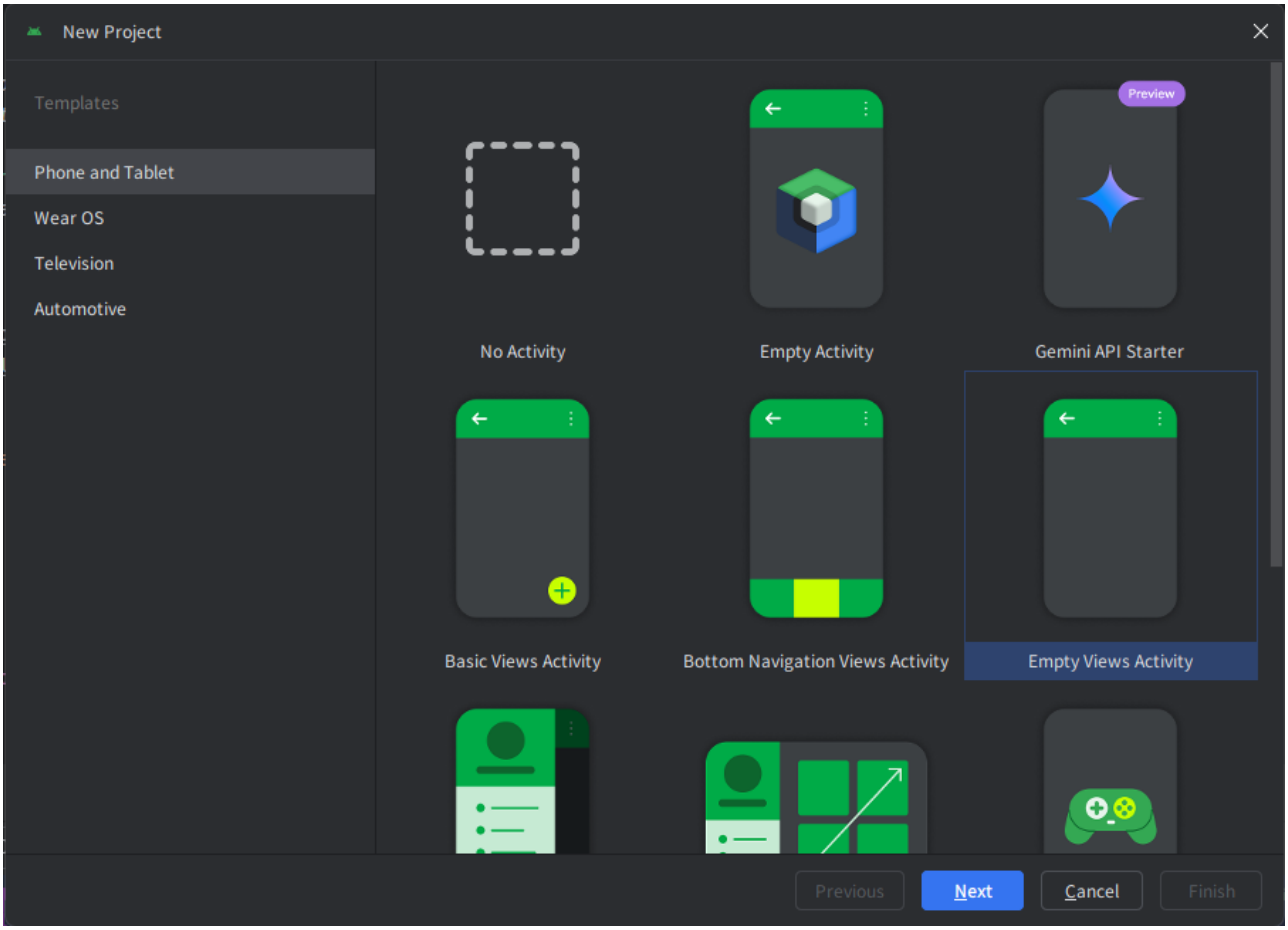

## 2.3 How to Run the Demo

The ComPDFKit Conversion SDK for Android provides a Kotlin demo sample to help developers understand how to use the SDK on Android. You can find this demo sample in the ***"samples"*** folder. This guide uses the Kotlin demo to show how to run it in Android Studio.

## 2.4 How to Create an Android Application

### 2.4.1 Create a New Project

1. Open Android Studio, then click **File** -> **New** -> **New Project**. As shown in Figure 2-1, select **Empty Views Activity**.




2. Choose the project name, package name, desired location, and the corresponding minimum SDK version, then **click Finish**.

## 2.4.2 Add the ComPDFKit Conversion SDK to Your Project

1. Copy all files from the "***lib***" folder to your project's "***app/main***" directory.
2. Add the following line to the `dependencies` block in "***app/build.gradle.kts***":

```
...

dependencies {
    ...
    implementation(files("lib/ComPDFKitConversion-release.aar"))
    ...
}
```

3. Sync the Gradle configuration to ensure the ComPDFKit Conversion SDK is added to the project.

### 2.5 Applying the License

Before using the classes and methods of the ComPDFKit Conversion SDK in your project, you need to initialize the SDK with a valid license key. If you don't have a license key, feel free to contact the ComPDFKit team to obtain one.

Add the following code inside the `<application>` tag of your module's AndroidManifest.xml to apply the license:

```
<meta-data
    android:name="conversion_license"
    android:value="..." />
```

## 3. Conversion Guides

---

ComPDFKit Conversion SDK allows developers to use a simple API to convert PDF to the most commonly used file formats like Word, Excel, PPT, HTML, CSV, PNG, JPEG, RTF, CSV,etc. Provide a wealth of customized conversion options, such as whether to include images or annotations in PDF documents, whether to enable OCR or layout analysis, etc.

### 3.1 Get Conversion Progress

ComPDFKit Conversion SDK obtains the conversion progress through callback functions. The following example demonstrates how to get the conversion progress while performing a PDF to Word task:

In the SDK, the following interfaces are available:

```
interface ProgressCallback {
    fun onProgress(current: Int, total: Int)
}
```

Implement this interface in your class and override the following function:

```
class ConversionTask(
    var path: String,
    var outputPath: String,
    var options: WordOptions
) : ProgressCallback {
```

```
    override fun onProgress(current: Int, total: Int) {
        // Add the callback function code at this location.
    }

    fun startTask() {
        ConverterManager.setProgress(this)
        val errorCode = ComPDFKitConverter.startPDFToWord(path, "",
outputPath, options)
    }
}
```

## 3.2 Cancel Conversion Task

ComPDFKit Conversion SDK supports interrupting your ongoing conversion task at any time. The following example demonstrates how to interrupt your ongoing conversion task:

```
ComPDFKitConverter.cancel()
```

## 3.3 Select Page Range for Conversion

ComPDFKit Conversion SDK supports converting a specific page range. If an empty string is passed, it will convert all pages. Below is an example showing how to specify a page range during conversion:

```
val wordOptions = WordOptions();
wordOptions.pageRanges = "1-3,5,7-9";
```

## 3.4 Contain Image and Annotation Options

### Overview

In the process of converting PDF documents into various formats, ComPDFKit Conversion SDK offers two additional options for users: one option to determine whether images are included in the generated document, and another to decide if annotations from the PDF file are to be retained.

- With the "Include Images" option enabled, ComPDFKit Conversion SDK will extract the images from the PDF document and embed them in the

corresponding pages and positions in the output file. For areas with overlapping images, ComPDFKit Conversion SDK merges these images into one and embeds it into the exact location on the corresponding page of the output file.
- When the "Include Annotations" option is selected, most annotations are converted into raster images and embedded at the respective positions within your document. However, certain types of annotations, such as highlights, underlines, strikeouts, and squiggly, are converted into their respective formatting equivalents in the converted Word, PPT, and HTML documents, and are marked over the corresponding text. It is important to note that the conversion won't be 100% accurate in every instance.

In the ComPDFKit Conversion SDK, the options of including image and annotation are commonly used in the following format conversion:

- PDF to Word
- PDF to Excel
- PDF to PPT
- PDF to HTML
- PDF to RTF
- PDF to JSON
- PDF to Markdown

## About Text Markup Annotation

- **Highlight Annotation:** When converting to Word format, Microsoft Word only supports 15 highlight colors. To best replicate the original document, the highlighted text will have a background color that matches the original annotation’s color. In PPT, native highlight tags are used for the corresponding marked text. In HTML format, a `<span>` tag is created around the highlighted text, and the background color is set to match the original annotation.
- **Underline & Wavy Line Annotations:** When converting to Word and PPT formats, underlined or wavy lines will appear over the marked text. In HTML, the corresponding styles are applied to represent these annotations. If a piece of text is marked with both underline and wavy lines, only one will be applied during conversion, as underlines are essentially a form of wavy lines in Word, PPT, and HTML.
- **Strikethrough Annotations:** When converting to Word and PPT formats, strikethroughs will be applied over the marked text. However, the color of the strikethrough may not match the original PDF, as Word and PPT rely on the font color. In HTML, the strikethrough will maintain the original color.

## Sample

This Sample demonstrates how to use the ComPDFKit Conversion SDK to convert a PDF document to a Word document with the selected options: Include images and annotations.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
wordOptions.containImage = true;
wordOptions.containAnnotation = true;
wordOptions.enableAiLayout = true;
wordOptions.enableOcr = false;

val error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);
```

## 3.5 Page Layout Modes

In certain formats, the page layout mode plays a key role in the quality of the converted document. ComPDFKit Conversion SDK supports two layout modes: Flow Layout and Box Layout.

- **Flow Layout:** This layout uses paragraph indentations, columns, and tab positions to adjust the content. Its main advantage is flexibility; content can flow automatically as the document is edited, and it adapts to various screen sizes on different devices. This layout also supports structured maintenance and can implement consistent global formatting through style templates (e.g., titles, body text). Common use cases include documents that are frequently modified, such as reports, manuals, and dynamic tables.
- **Box Layout:** Based on the PDF's "digital paper" model, this layout accurately positions every element (text, images, tables) on the page using a coordinate system (e.g., text is positioned 5 cm from the top and 3 cm from the left). The main advantage is high-precision rendering, which ensures consistency across different platforms. This layout is particularly useful for documents requiring precise reproduction, such as contracts, design drafts, and academic papers.

In the ComPDFKit Conversion SDK, page layout modes are commonly used in the following format conversions:

- PDF to Word

- PDF to HTML

## Sample

This example demonstrates how to convert a PDF document to Word with Flow Layout and Box Layout:

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
wordOptions.layoutMode = PageLayoutMode.FLOW;
var error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);

wordOptions.layoutMode = PageLayoutMode.BOX;
error = ComPDFKitConverter.startPDFToWord(inputFilePath, password,
outputFileName, wordOptions);
```

# 3.6 OCR

## Overview

OCR (Optical Character Recognition) is the process of converting images of typed, handwritten, or printed text into machine-encoded text.

OCR is commonly used for text recognition and extraction from the following types of documents:

- Non-editable scanned PDF files.
- Photographs of documents.
- Scene photos such as advertising layouts, signboards, etc.
- Identification cards, passports, vehicle license plates, and other official plates.
- Invoices, bills, receipts, and other financial documents.

The following features support OCR:

- PDF to Word
- PDF to Excel
- PDF to PPT

- PDF to HTML
- PDF to RTF
- PDF to TXT
- PDF to CSV
- PDF to JSON
- PDF to Markdown

OCR Language Support of ComPDFKit Conversion SDK:

| SCRIPT / NOTICE | LANGUAGE (NATIVE) | LANGUAGE (IN ENGLISH) |
|---|---|---|
| Latn; American | English | English |
| Latn; Canadian | Français canadien | French |
| Hans/Hant | 中文简体 | Chinese (Simplified) |
| Hans/Hant | 中文繁体 | Chinese (Traditional) |
| Jpan | 日本語 | Japanese |
| Kore | 한국어 | Korean |
| Latn | Deutsch | German |
| Latn | Српски (латиница) | Serbian (latin) |
| Latn | Occitan, lenga d'òc, provençal | Occitan |
| Latn | Dansk | Danish |
| Latn | Italiano | Italian |
| Latn; European | Español | Spanish |
| Latn; European | Português (Portugal) | Portuguese |
| Latn | Te reo Māori | Maori |
| Latn | Bahasa Melayu | Malay |
| Latn | Malti | Maltese |
| Latn | Nederlands | Dutch |
| Latn; Bokmål | Norsk | Norwegian |
| Latn | Polski | Polish |
| Latn | Română | Romanian |
| Latn | Slovenčina | Slovak |
| Latn | Slovenščina | Slovenian |
| Latn | shqip | Albanian |
| Latn | Svenska | Swedish |

| SCRIPT / NOTICE | LANGUAGE (NATIVE) | LANGUAGE (IN ENGLISH) |
|---|---|---|
| Latn | Swahili | Swahili |
| Latn | Wikang Tagalog | Tagalog |
| Latn | Türkçe | Turkish |
| Latn | o‘zbekcha | Uzbek |
| Latn | Tiếng Việt | Vietnamese |
| Latn | Afrikaans | Afrikaans |
| Latn | Azərbaycan | Azerbaijani |
| Latn | Bosanski | Bosnian |
| Latn | Čeština | Czech |
| Latn | Cymraeg | Welsh |
| Latn | Eesti keel | Estonian |
| Latn | Gaeilge | Irish |
| Latn | Hrvatski | Croatian |
| Latn | Magyar | Hungarian |
| Latn | Bahasa Indonesia | Indonesian |
| Latn | Íslenska | Icelandic |
| Latn | Kurdî | Kurdish |
| Latn | Lietuvių | Lithuanian |
| Latn | Latviešu | Latvian |

## Converting Images to Other Document Formats

The OCR function also supports converting input images into Word, Excel, PPT, HTML, CSV, RTF, TXT, Json and other formats. This sample demonstrates how to use the ComPDFKit OCR function to convert image files to DOCX file.

```
val modelPath = "***";
ConverterManager.setAIModel(modelPath, OCRLanguage.CHINESE);

// Support jpg, jpeg, png, bmp, tiff format.
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
```

```
wordOptions.containImages = true;
wordOptions.containAnnotations = true;
wordOptions.enableAiLayout = true;
wordOptions.enableOcr = true;

var error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);
```

## Notice

- The quality of the OCR result depends on the quality of the input image. If the input image has a low resolution, the OCR result quality will be affected. A good rule of thumb is that the more pixels in the character shapes, the better. If the character bounding box is smaller than 20x20 pixels, OCR quality will drop exponentially. The ideal image is a grayscale image with a resolution around 300 DPI.
- When performing OCR, make sure the OCR language setting matches the language in the PDF document to achieve the best OCR conversion quality.
- OCR functionality currently does not support operating systems lower than Windows 10.

## Sample

This Sample demonstrates how to use the ComPDFKit OCR function to convert a PDF to DOCX file.

```
val modelPath = "***";
ConverterManager.setAIModel(modelPath, OCRLanguage.CHINESE);

// Support jpg, jpeg, png, bmp, tiff format.
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
wordOptions.containImages = true;
wordOptions.containAnnotations = true;
wordOptions.enableAiLayout = true;
wordOptions.enableOcr = true;

val error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);
```

# 3.7 Layout Analysis

## Overview

Layout analysis is the process of leveraging Artificial Intelligence (AI) technology to parse and understand the structure of a document's layout. Its primary goal is to extract text, images, tables, layers, and other data from the input documents.

Layout analysis has several common use cases, including:

- Intelligent recognition of tables within PDF documents: This feature is particularly useful for analyzing company financial statements, invoices, bank statements, experimental data, medical test reports, and more.
- Smart extraction of text, images, or tables from PDF documents through layout analysis: This functionality greatly aids in the analysis and extraction of information from identification cards, receipts, licenses, documents, ancient books, and other various types of files.

Features that support Layout Analysis:

- PDF to Word
- PDF to Excel
- PDF to PPT
- PDF to HTML
- PDF to RTF
- PDF to TXT
- PDF to CSV
- PDF to JSON
- PDF to Markdown

## Notice

- You need to integrate the OCR module before using layout analysis.
- When the OCR is enabled, the layout analysis is automatically enabled.

## Sample

This Sample demonstrates how to use the ComPDFKit OCR function to convert PDF to DOCX file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
wordOptions.containImages = true;
wordOptions.containAnnotations = true;
wordOptions.enableAiLayout = true;
wordOptions.enableOCR = false;

val error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);
```

## 3.8 Convert PDF to Word

### Overview

Converting PDF to Word is an operation that converts the PDF format file into a editing Word format file. By converting PDF to Word, you can easily edit, modify, insert, or delete text and pictures, adjust layout and properties.

### Layout differences

- **Word's Streaming Layout** Ideal for editing, with your editing, the content dynamically adapts to different positions. However, a Word file would display differently due to the incompatibility of various software or app versions. It makes it unsuitable for precise documentation like electronic files or certificates.
- **PDF's Fixed Page Layout:** Ensures a stable, uniform appearance and print quality across all devices. The content and formatting are locked upon creation, making alterations difficult without affecting the overall layout. It's preferred for formal documentation such as business reports and official electronic records.

### Sample

This sample demonstrates how to convert from a PDF to DOCX file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val wordOptions = WordOptions();
val error = ComPDFKitConverter.startPDFToWord(inputFilePath,
password, outputFileName, wordOptions);
```

## 3.9 Convert PDF to Excel

### Overview

ComPDFKit Conversion SDK supports converting PDF documents to Microsoft Excel format (.xlsx). By extracting, parsing, and importing data from PDF into Excel, users can further edit, analyze, or share Excel files. This feature helps increase productivity, reduce manual entry errors, and simplify complex document processing tasks.

### Set the content options for Excel

When converting PDF files to Excel files, you need to pay attention to the settings of the following options, which will directly affect the content written to the Excel file.

- Content options:
  If you call the `setAllContent(true)`, The converted Xlsx file will contain all the contents in the PDF.
- Worksheet options:

| OPTION | DESCRIPTION |
| --- | --- |
| `FOR_TABLE` | Create one sheet for one table. |
| `FOR_PAGE` | Create one sheet for one PDF page. |
| `FOR_DOCUMENT` | Create one sheet for the entire PDF document. |

### Sample

This sample demonstrates how to convert from a PDF to XLSX file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val excelOptions = ExcelOptions();
val error = ComPDFKitConverter.startPDFToExcel(inputFilePath,
password, outputFileName, excelOptions);
```

## 3.10 Convert PDF to PowerPoint

### Overview

ComPDFKit Conversion SDK provides the function of converting PDF files to PowerPoint files and restoring the layout and format of the original document, which can meet the needs of users for the presentation and editing of document content in Microsoft PowerPoint.

### Sample

This sample demonstrates how to convert from a PDF to PPTX file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val pptOptions = PptOptions();
val error = ComPDFKitConverter.startPDFToPpt(inputFilePath,
password, outputFileName, pptOptions);
```

## 3.11 Convert PDF to HTML

### Overview

ComPDFKit Conversion SDK provides the PDF to HTML function, which can convert PDF files to HTML files while maintaining the layout and format of the original document, allowing users to browse and view the document on Web.

## Notice

When converting PDF to HTML format, ComPDFKit Conversion SDK provides the following four options to create HTML files:

| OPTIONS | DESCRIPTION |
| --- | --- |
| `SINGLE_PAGE` | Convert the entire PDF file into a single HTML file, where all PDF pages are connected in sequence according to page number, displayed on the same HTML page. |
| `SINGLE_PAGE_WITH_BOOKMARK` | Convert the PDF file into a single HTML file with an outline for navigation at the beginning of the HTML page. Still, all PDF pages are connected in sequence according to page number, displayed on the same HTML page. |
| `MULTIPLE_PAGE` | Convert the PDF file into multiple HTML files. Each HTML file corresponds to a PDF page, and users can navigate to the next HTML file via a link at the bottom of the HTML page. |
| `MULTIPLE_PAGE_WITH_BOOKMARK` | Convert the PDF file into multiple HTML files. Each HTML file corresponds to a PDF page, and users can navigate to the next HTML file via a link at the bottom of the HTML page. The links of all the HTML files are presented in an outline HTML file for navigation. |

## Sample

This sample demonstrates how to convert from a PDF to HTML file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val htmlOptions = HtmlOptions();
val error = ComPDFKitConverter.startPDFToHtml(inputFilePath,
password, outputFileName, htmlOptions);
```

## 3.12 Convert PDF to CSV

### Overview

ComPDFKit Conversion SDK supports converting PDF documents to CSV (Comma-Separated Values). Converting PDF to CSV is a common need, usually used to extract tabular or structured data from PDF documents and convert them into CSV files.

### Sample

This sample demonstrates how to convert from a PDF to CSV file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val excelOptions = ExcelOptions();
excelOptions.csvFormat = true;
val error = ComPDFKitConverter.startPDFToExcel(inputFilePath,
password, outputFileName, excelOptions);
```

## 3.13 Convert PDF to Image

### Overview

ComPDFKit Conversion SDK provides an API for converting PDF to images. Integrate ComPDFKit Conversion SDK to your apps to convert PDF into images easily.

### Setting Image Formats

In ComPDFKit Conversion SDK, supported image formats include:

- JPG
- JPEG
- PNG
- BMP
- TIFF

## Setting Image Color Modes

Supported image color modes in ComPDFKit Conversion SDK include:

- **Color:** Color mode, where the image effect is consistent with the original PDF page.
- **Gray:** Grayscale mode.
- **Binary:** Black and white mode, which applies binarization to the original effect.

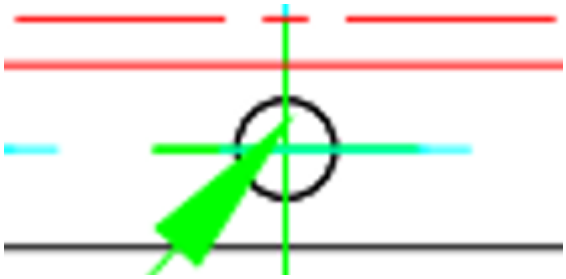
## Setting Image Scaling

The SDK supports setting image scaling. The default scaling is 1.0, which maintains the original PDF page size. If you want to double the image size, you can `setImageScaling` to 2.0; similarly, to reduce the image size by half, `setImageScaling` to 0.5.

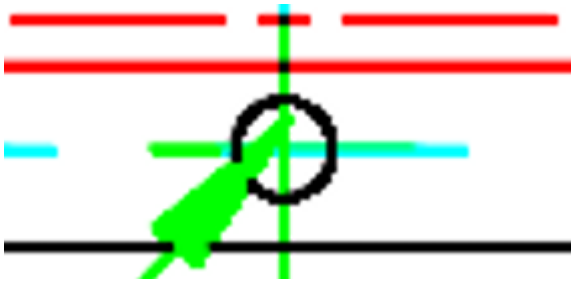
## Enhancing Image Path Display

The SDK supports an option called `image_path_enhance` for enhancing the display of image paths. This option can be enabled when you want to enhance the display effect of paths within the PDF page.

- Not enable `image_path_enhance` option(original PDF rendering effect):



- Enable `image_path_enhance` option:



## Notice

- A higher `imageScaling` value results in images with higher resolution, but it also increases memory usage and slows down the conversion.
- A higher `imageScaling` value does not necessarily equate to higher clarity; the clarity also depends on the original image resolution in the document.

## Sample

The following complete example code demonstrates how to convert a PDF document into PNG format.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val imageOptions = ImageOptions();
val error = ComPDFKitConverter.startPDFToImage(inputFilePath,
password, outputFileName, imageOptions);
```

# 3.14 Convert PDF to RTF

## Overview

RTF is a popular text format that can retain the format and style data of the text, and it is convenient for most text readers to read and write. Integrate ComPDFKit Conversion SDK to convert PDF to RTF files now.

## Sample

This sample demonstrates how to convert from a PDF to RTF file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val rtfOptions = RtfOptions();
val error = ComPDFKitConverter.startPDFToRtf(inputFilePath,
password, outputFileName, rtfOptions);
```

# 3.15 Convert PDF to TXT

## Overview

When you need to extract the text content in the PDF file, in order for data analysis, text mining, information retrieval, etc. Using ComPDFKit Convert SDK, you can easily extract the text in the PDF into the TXT file.

## Preserving Table Format

The SDK supports a function called `setTableFormat()` that preserves the table format when writing the TXT file, meaning that the original table structure is maintained. It is generally recommended to enable this option, especially useful for data extraction scenarios.

## Sample

This sample demonstrates how to convert from a PDF to TXT file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val txtOptions = TxtOptions();
val error = ComPDFKitConverter.startPDFToTxt(inputFilePath,
password, outputFileName, txtOptions);
```

# 3.16 Convert PDF to Searchable PDF

## Overview

To make a searchable PDF by adding invisible text to an image based PDF such as a scanned document using OCR.

## Sample

Full code sample which shows how to use the ComPDFKit OCR module on scanned documents in multiple languages.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val pdfOptions = PdfOptions();
pdfOptions.enableOcr = true;
val error =
ComPDFKitConverter.startPDFToSearchablePDF(inputFilePath, password,
outputFileName, pdfOptions);
```

### 3.18 Releasing Library Resources

#### Overview

Releases the files and memory resources occupied by the ComPDFKit Conversion SDK.

#### Notice

- After calling this interface to release library resources, the ComPDFKit Conversion SDK will no longer function properly and must be reloaded.

#### Sample

```
ConverterManager.release();
```

# 4 Data Extraction Guide

---

Unleash the Power of Data with ComPDFKit Conversion SDK's Data Extraction to detect, recognize, analyze, and extract the PDF text, image, table, etc.

### 4.1 Extract PDF To Json

## Overview

Extract text, tables and images from PDF documents to Json file.

## Standard table and non-standard table

Commonly, tables can be divided into two categories: standard tables and non-standard tables. The specific definitions are as follows:

- Standard table: The table border and the inner lines of the table are complete and clear. There is no need to manually add table lines to divide the table content.

| EXTERIOR - RAMPS, DOORS, and LIFTS | | | |
|---|---|---|---|
| Spec # | Spec | Initials | Notes |
| 101 | **REMOVAL OF ITEMS FROM WORK AREA**<br>Homeowner is responsible for removing objects from the work area. However, contractor is to double check that the items have been removed and that the area is prepared for conversion. | | |
| 102 | **INITIAL INSPECTION AND REPAIR**<br>Inspect the general area (concrete, doorway, threshold, etc.) for damage. Immediately report any "unforeseen items" to the Care Manager and Koremen. | | |
| 103 | **PREPARE YARD**<br>Prepare the existing yard for the ramp modification and cement. This includes the addition of dirt as needed, the excavating of the land, leveling, and grading of soil away from the home. | | |
| 104 | **DEMOLITION**<br>Demolish and remove existing areas *only as necessary* to construct the modification outlined in the specifications. Dispose of tear out in a code legal dump. Follow all environmental protection measures (lead safe practices) as required by the EPA and local ordinances. | | Existing landing and ramps |
| 105A | **DOOR THRESHOLD LANDING FOR VPL**<br>Construct a preservative treated wood landing at the door threshold height, approximately 42" above ground level or the concrete. 5'x6'. Add new steps. The landing should attach to existing porch and connect to new VPL.<br>(Refer to diagram) | | McKinley Collins approved aluminum modular platform and stairs with hand rails. |
| 106 | **VERTICAL PLATFORM LIFT**<br>Install a Bruno (or equivalent) vertical platform lift system per manufacturer's specifications. Ensure the lift has access to appropriate electrical outlets.<br><br>Lift to have battery backup and call capability from bottom or top. | | Pour a concrete base for the VPL. Ensure level. Must have manual override for power outages. Ensure smooth operation. |
| 106a | **ELECTRIC FOR INSTALLING VPL**<br>Check with local zoning, fire, electric, and other code enforcement agencies regarding contractor licensing required for installation. Install based upon these requirements. | | Add outlet to side of house for VPL. |
| 115 | **CONCRETE LANDING**<br>Excavate area to create new concrete landing and loading area. Form and pour 4" thick, 25SF, 2200 PSI concrete slab, to create an area to allow VPL stationary installation and client access point. Use straight, solid forms between temps of 40-100F. All concrete shall be:<br>o free of voids and cavities.<br>o treated with liquid curing compound.<br>o protected from the weather while curing.<br>o broom finished across direction of traffic.<br><br>Create a smooth transition onto the driveway/sidewalk from landing. Remove all forms, re-grade and spot seed. | | Extend the concrete a minimum 5' wide to the edge of the home. |

- **Non-Standard Tables:** Tables lacking borders or clear inner lines, requiring manual additions of table lines to separate contents.

| features | precision | recall | F1-score | MCC | AUROC |
|---|---|---|---|---|---|
| B | $0.823 \pm 0.311$ | $0.024 \pm 0.012$ | $0.047 \pm 0.022$ | $\mathbf{0.130} \pm 0.055$ | $0.799 \pm 0.013$ |
| BC | $0.818 \pm 0.173$ | $0.057 \pm 0.016$ | $0.106 \pm 0.028$ | $\mathbf{0.197} \pm 0.045$ | $0.842 \pm 0.005$ |
| BT | $0.542 \pm 0.177$ | $0.051 \pm 0.030$ | $0.092 \pm 0.052$ | $\mathbf{0.138} \pm 0.060$ | $0.786 \pm 0.009$ |
| BV | $0.745 \pm 0.040$ | $0.232 \pm 0.047$ | $0.350 \pm 0.068$ | $\mathbf{0.372} \pm 0.055$ | $0.815 \pm 0.006$ |
| BY | $0.781 \pm 0.022$ | $0.153 \pm 0.014$ | $0.256 \pm 0.020$ | $\mathbf{0.314} \pm 0.016$ | $0.820 \pm 0.004$ |
| BTV | $0.709 \pm 0.011$ | $0.268 \pm 0.013$ | $0.389 \pm 0.015$ | $\mathbf{0.393} \pm 0.013$ | $0.832 \pm 0.003$ |
| **BCTVY** | $\mathbf{0.793} \pm 0.009$ | $\mathbf{0.424} \pm 0.011$ | $\mathbf{0.552} \pm 0.010$ | $\mathbf{0.541} \pm 0.008$ | $\mathbf{0.890} \pm 0.002$ |

## Table Extraction Option

ComPDFKit Conversion SDK supports the function `setContainTable()`, when enabled, will extract table content from PDFs and output the table structure; otherwise, table content will be treated as regular text.

## Notice

- Without enabling AI layout analysis or OCR options, tables in the original PDF cannot be extracted. It is recommended to enable AI layout analysis or OCR for high-precision table recognition.

## Sample

Full sample code which illustrates the text extraction capabilities.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val jsonOptions = JsonOptions();
val error = ComPDFKitConverter.startPDFToJson(inputFilePath,
password, outputFileName, jsonOptions);
```

# 4.2 Convert PDF to Markdown

## Overview

To make a Markdown file from PDF content.

### Sample

Full code sample which shows how to convert from a PDF to Markdown file.

```
val inputFilePath = "***";
val password = "***";
val outputFileName = "***";

val markdownOptions = MarkdownOptions();
val error = ComPDFKitConverter.startPDFToMarkdown(inputFilePath,
password, outputFileName, jsonOptions);
```

# 5. Support

---

## 5.1 FAQ

- Does OCR work on x86 architecture?

  Currently, the OCR only works on x64 architecture.

## 5.2 Contact Us

Thanks for your interest in ComPDFKit Conversion SDK, the easy-to-use and powerful development solution. If you encounter technical questions or bug issues when using ComPDFKit Conversion SDK, please submit the problem report to the ComPDFKit team. More information as follows would help us to solve your problem:

- ComPDFKit Conversion SDK product and version.
- Your operating system and IDE version.
- Detailed descriptions of the problem.
- Any other related information, such as an error screenshot.

## Contact Inforation

- Home link: https://www.compdf.com
- Technical Support: https://www.compdf.com/support
- Email: support@compdf.com

Thanks,
The ComPDFKit Team