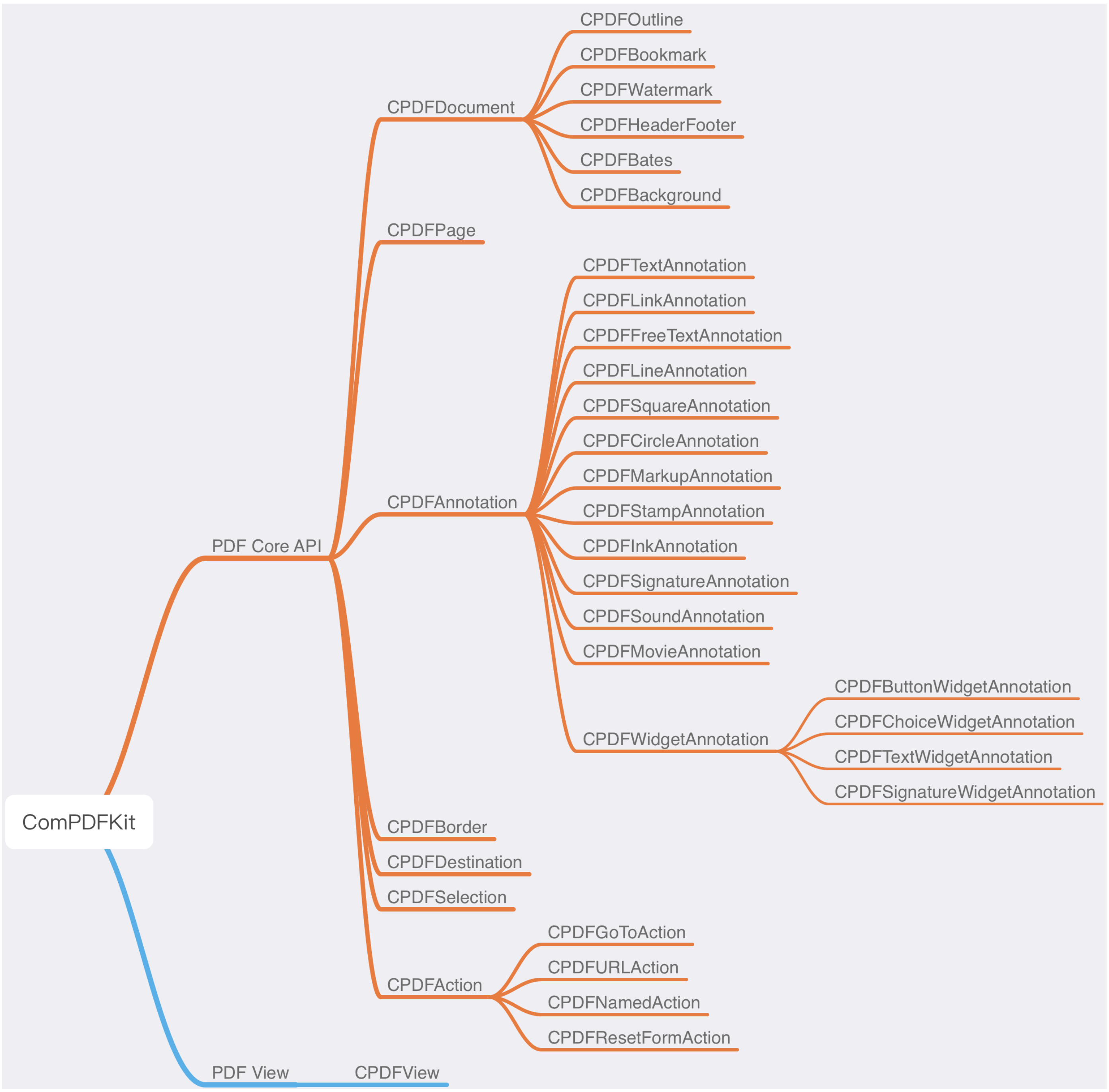
Contents

1 Overview

ComPDFKit PDF SDK for iOS is a robust PDF library for developers who need to develop applications on iOS, which offers powerful Objective-C APIs for quickly viewing, annotating, editing, and creating PDFs. It is feature-rich and battle-tested, making PDF files process and manipulation easier and faster for iOS devices.

1.1 ComPDFKit PDF SDK

ComPDFKit PDF SDK consists of two elements as shown in the following picture.



The two elements for ComPDFKit PDF SDK:

- **PDF Core API**

  The Core API can be used independently for document rendering, analysis, text extraction, text search, form filling, password security, annotation creation and manipulation, and much more.

- **PDF View**

  The PDF View is a utility class that provides the functionality for developers to interact with rendering PDF documents per their requirements. The View Control provides fast and high-quality rendering, zooming, scrolling, and page navigation features. The View Control is derived from platform-related viewer classes (e.g. `UIView` on iOS) and allows for extension to accommodate specific user needs.

1.2 Key Features

Viewer component offers:

- Standard page display modes, including Scrolling, Double page, Crop mode, and Cover mode.
- Navigation with thumbnails, outlines, and bookmarks.
- Text search & selection.
- Zoom in and out & Fit-page.
- Switch between different themes, including Dark mode, Sepia mode, Reseda mode, and Custom color mode.
- Text reflow.

Annotations component offers:

- Create, edit and remove annotations, including Note, Link, Freetext, Line, Square, Circle, Highlight, Underline, Squiggly, Strikeout, Stamp, Ink, Sound
- Support for annotation appearances.
- Import and export annotations to/from XFDF.
- Support for annotation flattening.
- Predefine annotations.

Forms component offers:

- Create, edit and remove form fields, including Push Button, Check Box, Radio Button, Text Field, Combo Box, List Box, and Signature.
- Fill PDF Forms.
- Support for PDF form flattening.

Document editor component offers:

- PDF manipulation, including Split pages, Extract pages, and Merge pages.
- Page edit, including Delete pages, Insert pages, Crop pages, Move pages, Rotate pages, Replace pages, and Exchange pages.
- Document information setting.
- Extract images.

Edit PDF component offers:

- Programmatically add and remove text in PDFs and make it possible to edit PDFs like Word. Allow selecting text to copy, resize, change colors, text alignment, and the position of text boxes.
- Undo or redo any change.

Security component offers:

- Encrypt and decrypt PDFs, including Permission setting and Password protected.
- Create, edit, and remove watermark.
- Redact content including images, text, and vector graphics.
- Create, edit, and remove header & footer, including dates, page numbers, document name, author name, and chapter name.
- Create, edit, and remove bates numbers.
- Create, edit, and remove background that can be a solid color or an image.

Conversion component offers:

- PDF to PDF/A.

1.3 License

ComPDFKit PDF SDK is a commercial SDK, which requires a license to grant developer permission to release their apps. Each license is only valid for one bundle ID in development mode. Other flexible licensing options are also supported, please contact our marketing team to know more. However, any documents, sample code, or source code distribution from the released package of ComPDFKit to any third party is prohibited.

2 Get Started

It is easy to embed ComPDFKit in your iOS app with a few lines of Objective-C code. Takes just a few minutes and gets started.

The following sections introduce the structure of the installation package, how to run a demo, and how to make an iOS app in Objective-C with ComPDFKit PDF SDK.

2.1 Requirements

ComPDFKit requires the latest stable version of Xcode available at the time the release was made. This is a hard requirement, as each version of Xcode is bundled with a specific version of the iOS Base SDK, which often defines how UIKit and various other frameworks behave.

- iOS 10.0 or higher.
- Xcode 12.0 or newer for Objective-C or Swift.

2.2 iOS Package Structure

The package of ComPDFKit PDF SDK for iOS includes the following files as shown in Figure 2-1:

- **ComPDFKit.xcframework** - Include the ComPDFKit dynamic library (arm64_armv7, x86_64-simulator) and associated header files.

- **PDFViewer** - A folder containing iOS sample projects.
- **PDFViewer-Swift** - A folder containing Swift iOS sample projects.
- **api_reference_ios** - API reference.
- **developer_guide_ios.pdf** - Developer guide.
- **release_notes.txt** - Release information.
- **legal.txt** - Legal and copyright information.

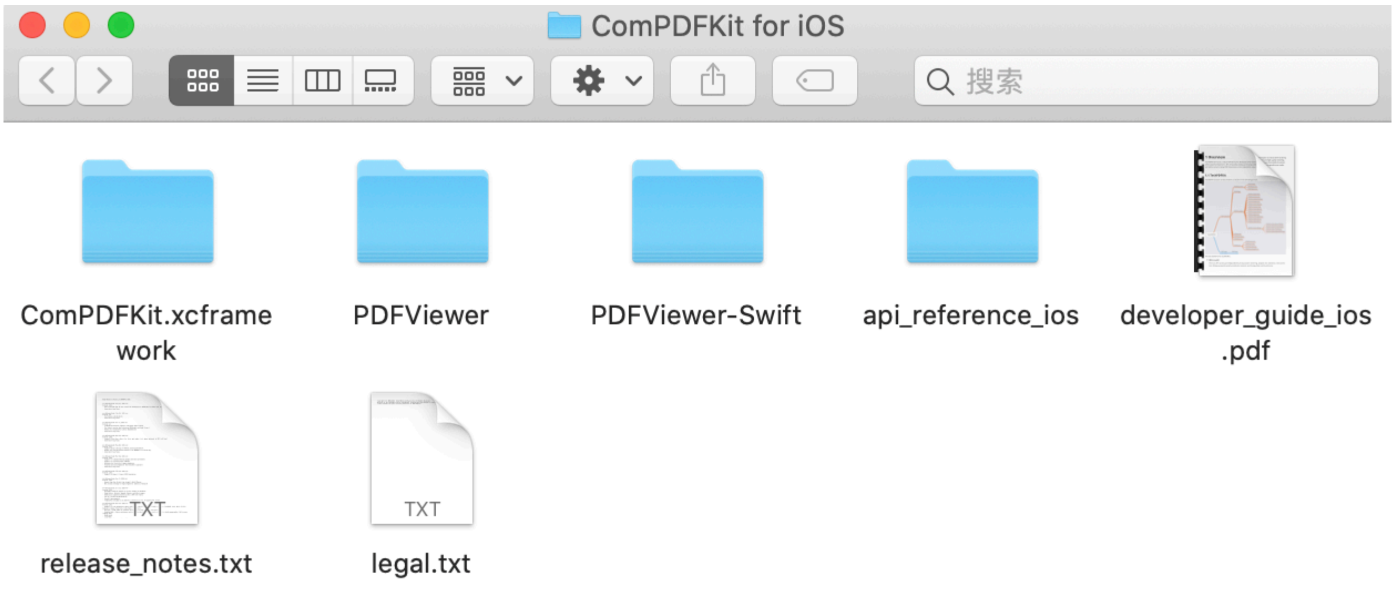
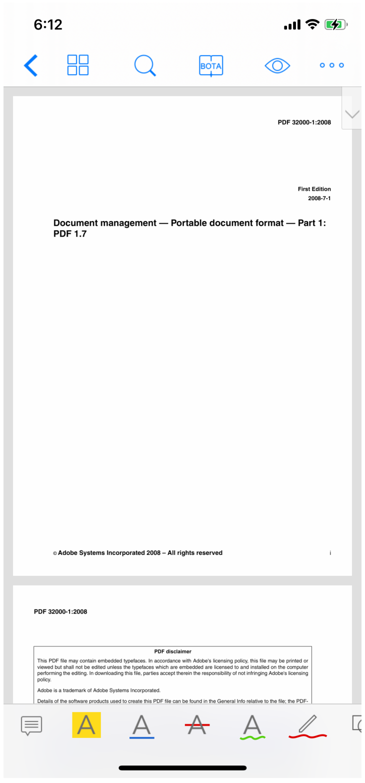


Figure 2-1

2.3 How to run a demo

ComPDFKit PDF SDK for iOS provides one demo in Objective-C for developers to learn how to call the SDK on iOS. You can find them in the ***"PDFViewer"*** folder. In this guide, it takes the "Objective-C" demo as an example to show how to run it in Xcode.

1. Double-click the ***"PDFViewer.xcodeproj"*** found in the ***"PDFViewer"*** folder to open the demo in Xcode.
2. Click on ***"Product -> Run"*** to run the demo on an iOS device. In this guide, an iPhone 7 Plus device will be used as an example. After building the demo successfully, on the start screen, click the ***"PDF32000_2008.pdf"*** document, and then it will be opened and displayed.



PDF 32000-1:2008

First Edition
2008-7-1

Document management — Portable document format — Part 1: PDF 1.7

© Adobe Systems Incorporated 2008 – All rights reserved i

PDF 32000-1:2008

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-

Note: *This is a demo project, presenting completed ComPDFKit PDF SDK functions. The functions might be different based on the license you have purchased. Please check that the functions you choose work fine in this demo project.*

2.4 How to Make an iOS App in Objective-C with ComPDFKit

This section will help you to quickly get started with ComPDFKit PDF SDK to make an iOS app in Objective-C with step-by-step instructions, which include the following steps:

1. Create a new iOS project in Objective-C.
2. Integrate ComPDFKit into your apps.
3. Apply the license key.
4. Display a PDF document.

2.4.1 Create a New iOS Project in Objective-C

In this guide, we use Xcode 12.4 to create a new iOS project.

Fire up Xcode, choose **File** -> **New** -> **Project...**, and then select **iOS** -> **Single View Application** as shown in Figure 2-2. Click **Next**.

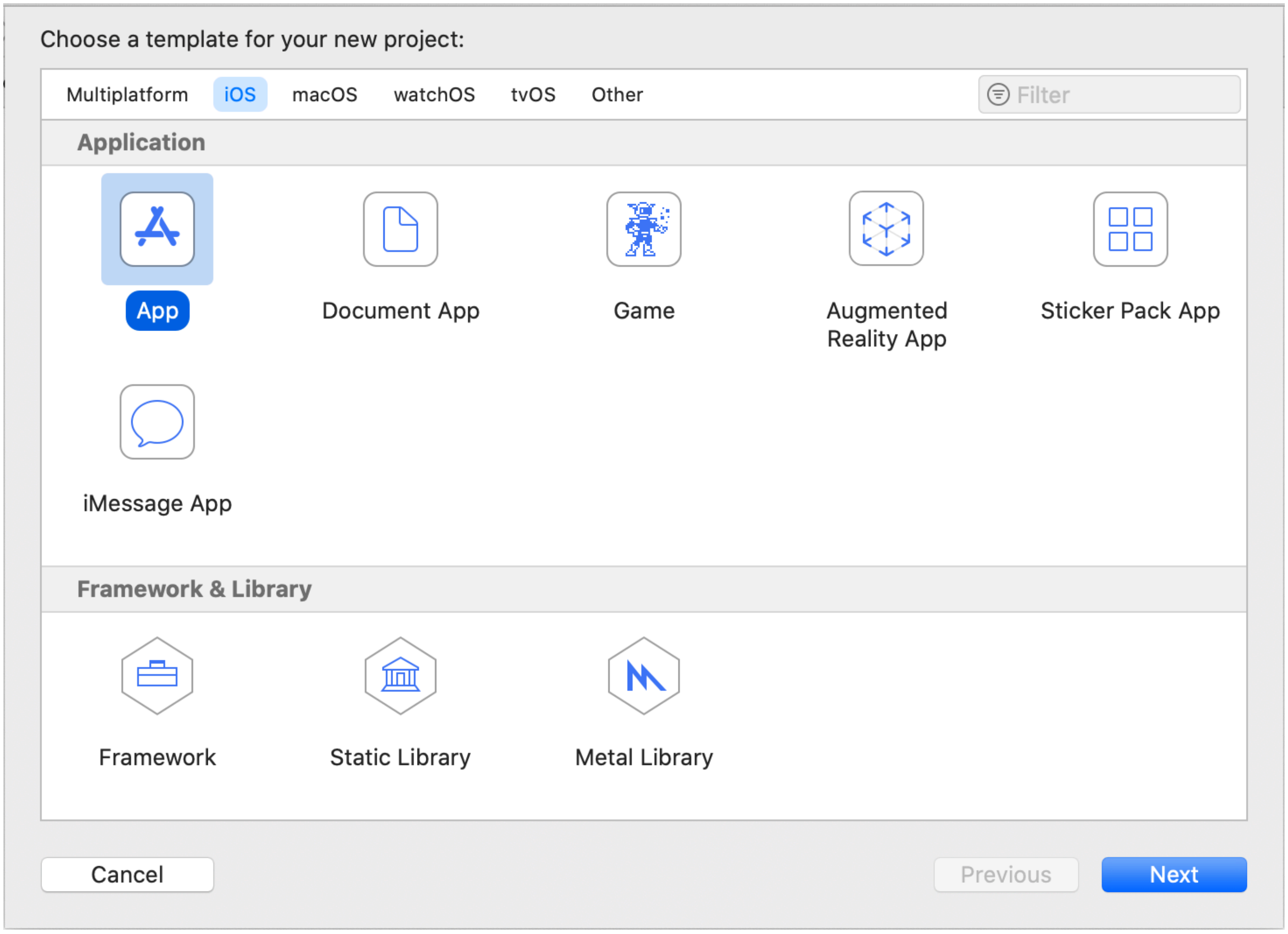
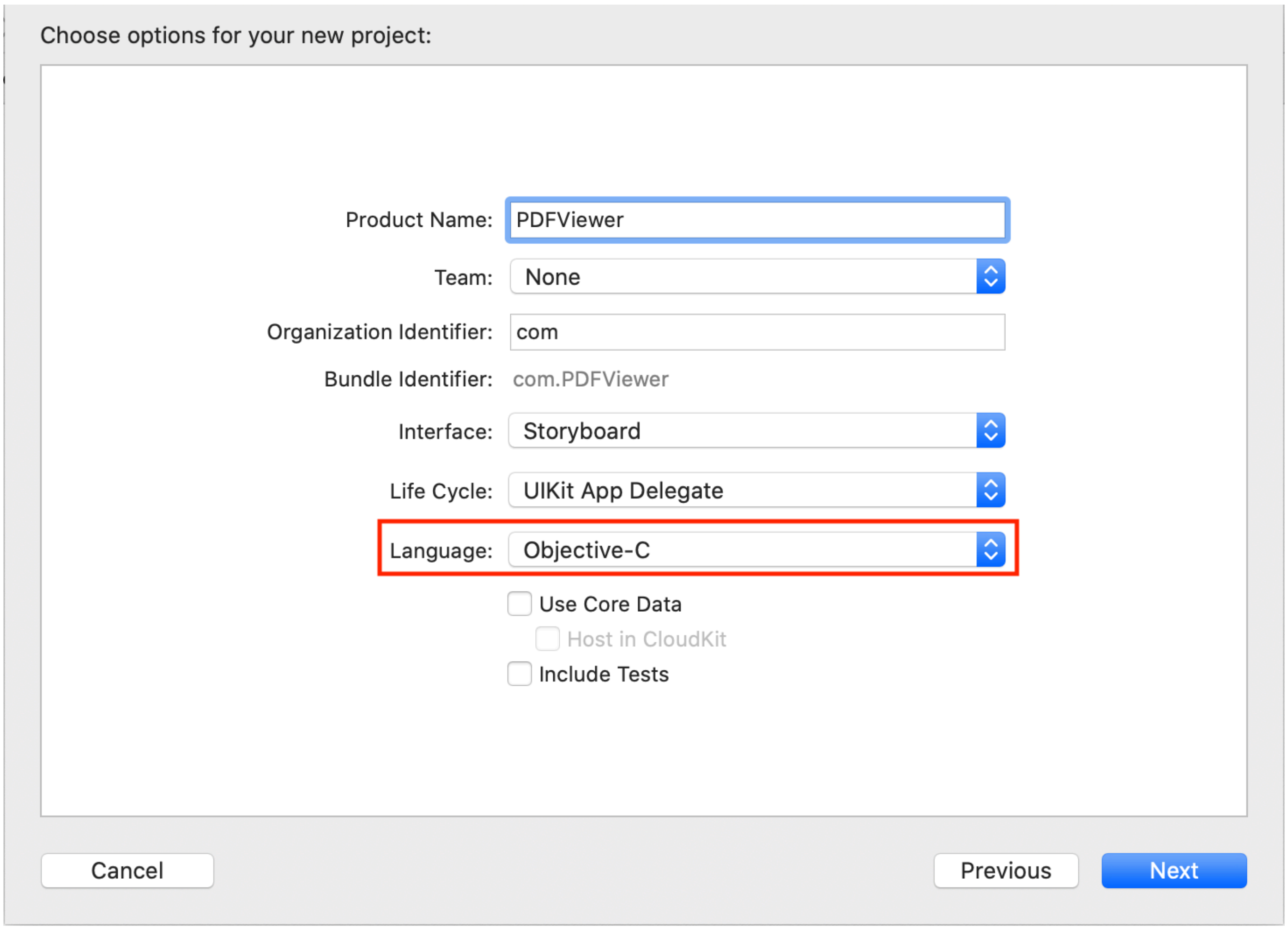


Figure 2-2

Choose the options for your new project as shown in Figure 2-3. Please make sure to choose Objective- C as the programming language. Then, click **Next**.


Choose options for your new project:

Product Name: PDFViewer

Team: None

Organization Identifier: com

Bundle Identifier: com.PDFViewer

Interface: Storyboard

Life Cycle: UIKit App Delegate

Language: Objective-C

Use Core Data [ ]

Host in CloudKit [ ]

Include Tests [ ]

Figure 2-3

Place the project to the location as desired. Then, click **Create**.

2.4.2 Integrate ComPDFKit into Your Apps

There are two ways to integrate ComPDFKit PDF SDK for iOS into your apps. You can choose what works best for you based on your requirements.

If you just want to use the default built-in UI implementations to develop your apps for simplicity and convenience, you need to include the following files:

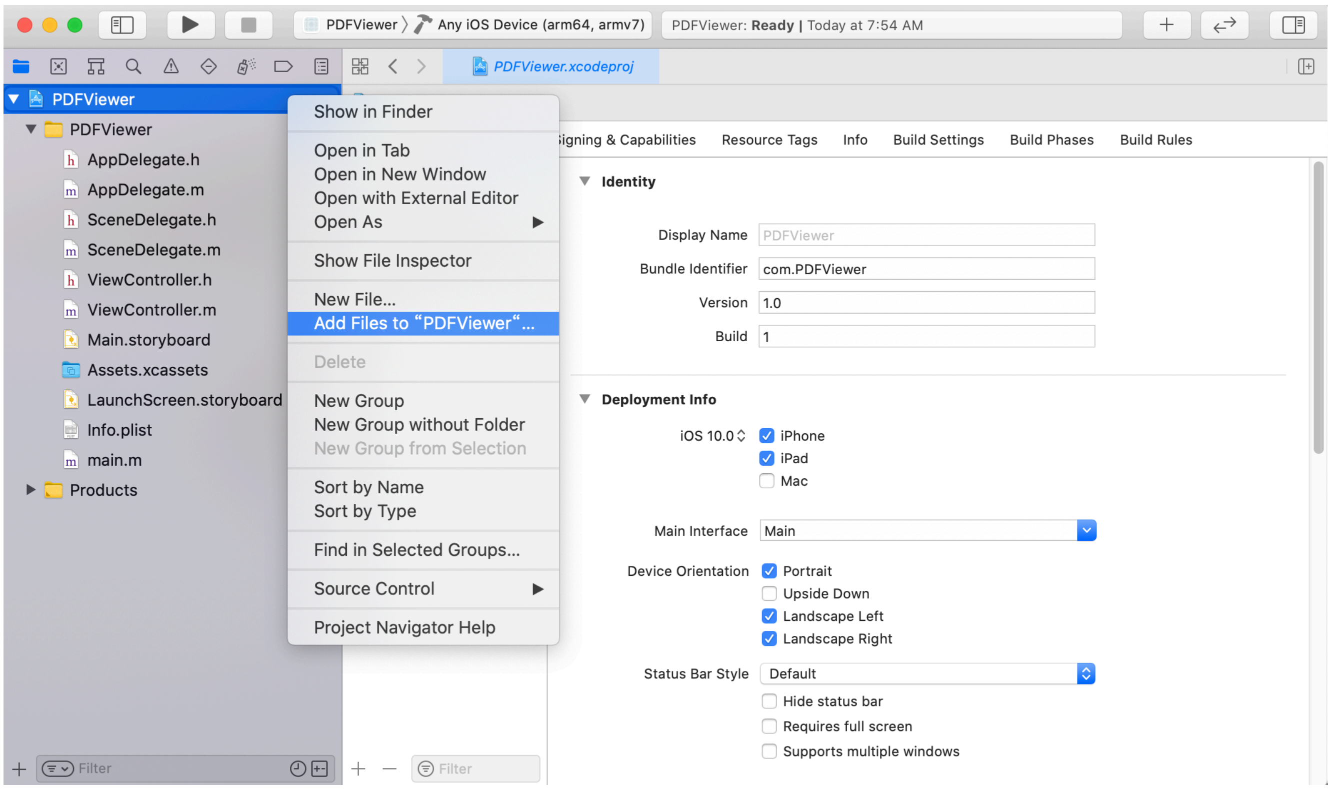
- **ComPDFKit.xcframework** - Include the ComPDFKit dynamic library (arm64_armv7, x86_64-simulator) and associated header files.
- **Source files** - Found in the **" *PDFViewer / Source* "** folder. They are the default built-in UI.
- **Resource files** - Found in the **" *PDFViewer / Resources* "** folder. They are needed for the default built-in UI implementations, such as images, strings, and other resources.

If you want to customize your (PDF-related) app's UI design, you need to include the following files:

- **ComPDFKit.xcframework** - Include the ComPDFKit PDF SDK dynamic library (arm64_armv7, x86_64-simulator) and associated header files.
- **Resource files** - Found in the **" *PDFViewer / Resources* "** folder. They are needed for the default built-in UI implementations, such as images, strings, and other resources.

To add the dynamic xcframework ***"ComPDFKit.xcframework"*** into the ***"PDFViewer"*** project, please follows the steps below:

1. Right-click the ***"PDFViewer"*** project, select **Add Files to "PDFViewer"...** as shown in Figure 2-4.



Identity

Display Name: PDFViewer
Bundle Identifier: com.PDFViewer
Version: 1.0
Build: 1

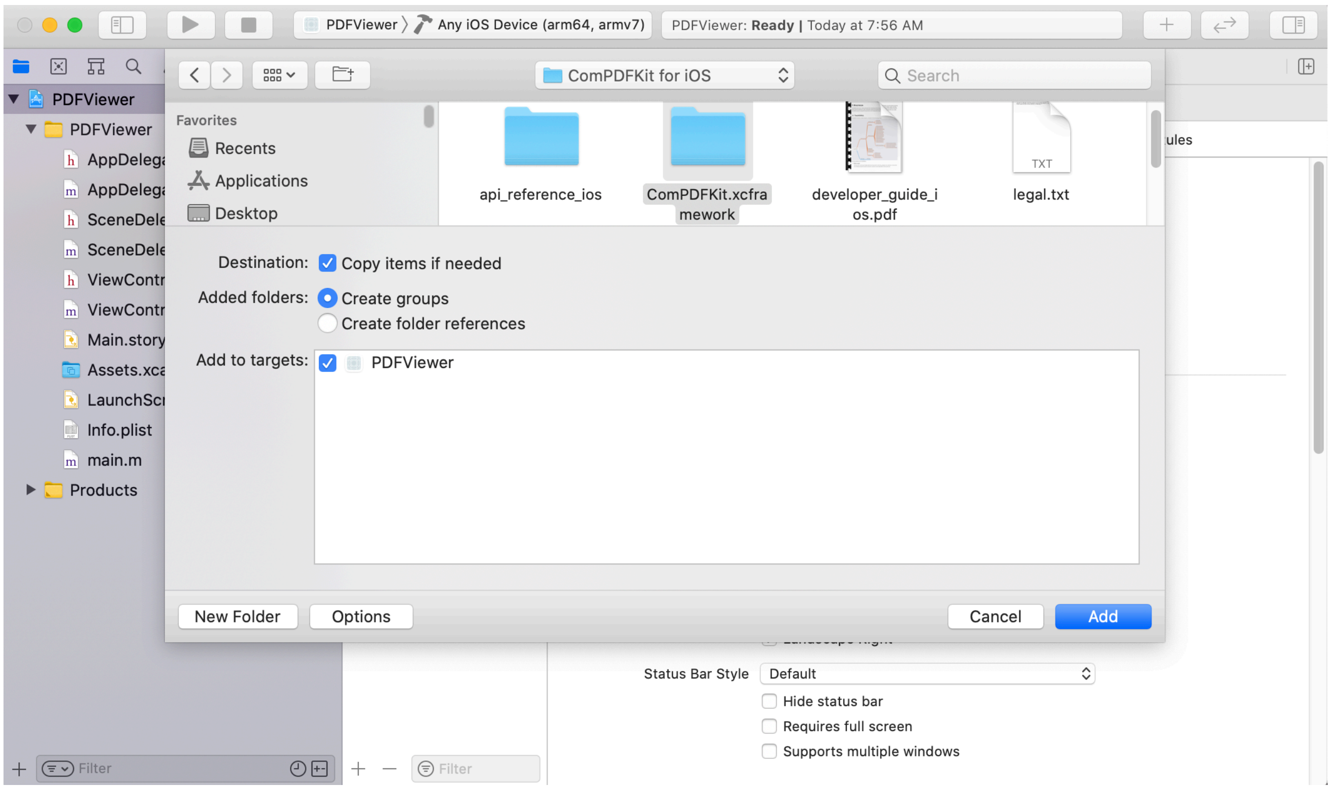
Deployment Info

iOS 10.0: ✓ iPhone, ✓ iPad, Mac
Main Interface: Main
Device Orientation: ✓ Portrait, Upside Down, ✓ Landscape Left, ✓ Landscape Right
Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-4

2. Find and choose ***"ComPDFKit.xcframework"*** in the download package, and then click **Add** as shown in Figure 2-5.

   Note: *Make sure to check the* ***"Copy items if needed"*** *option.*

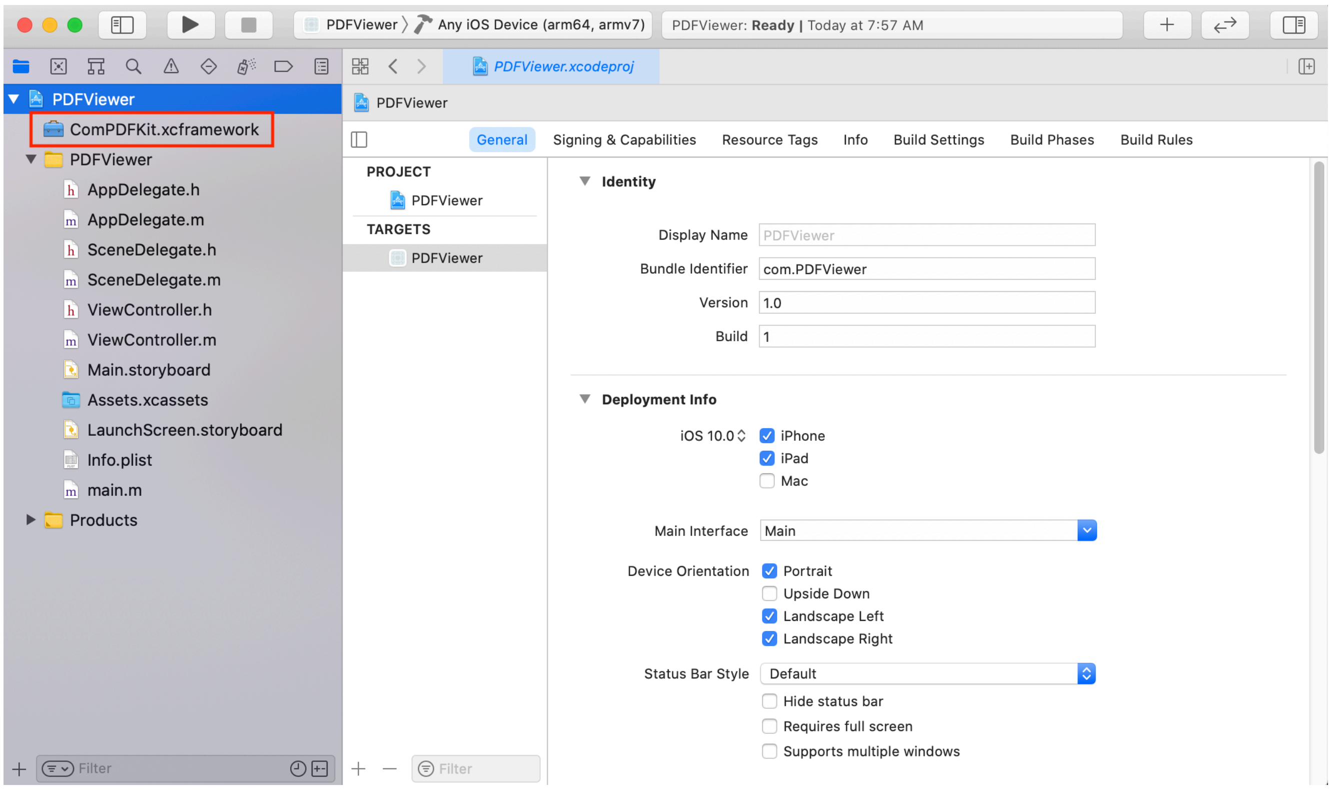


Destination: ✓ Copy items if needed
Added folders: [✓ Create groups] Create folder references
Add to targets: ✓ PDFViewer

Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-5

Then, the ***"PDFViewer"*** project will look like the Figure 2-6.



Identity

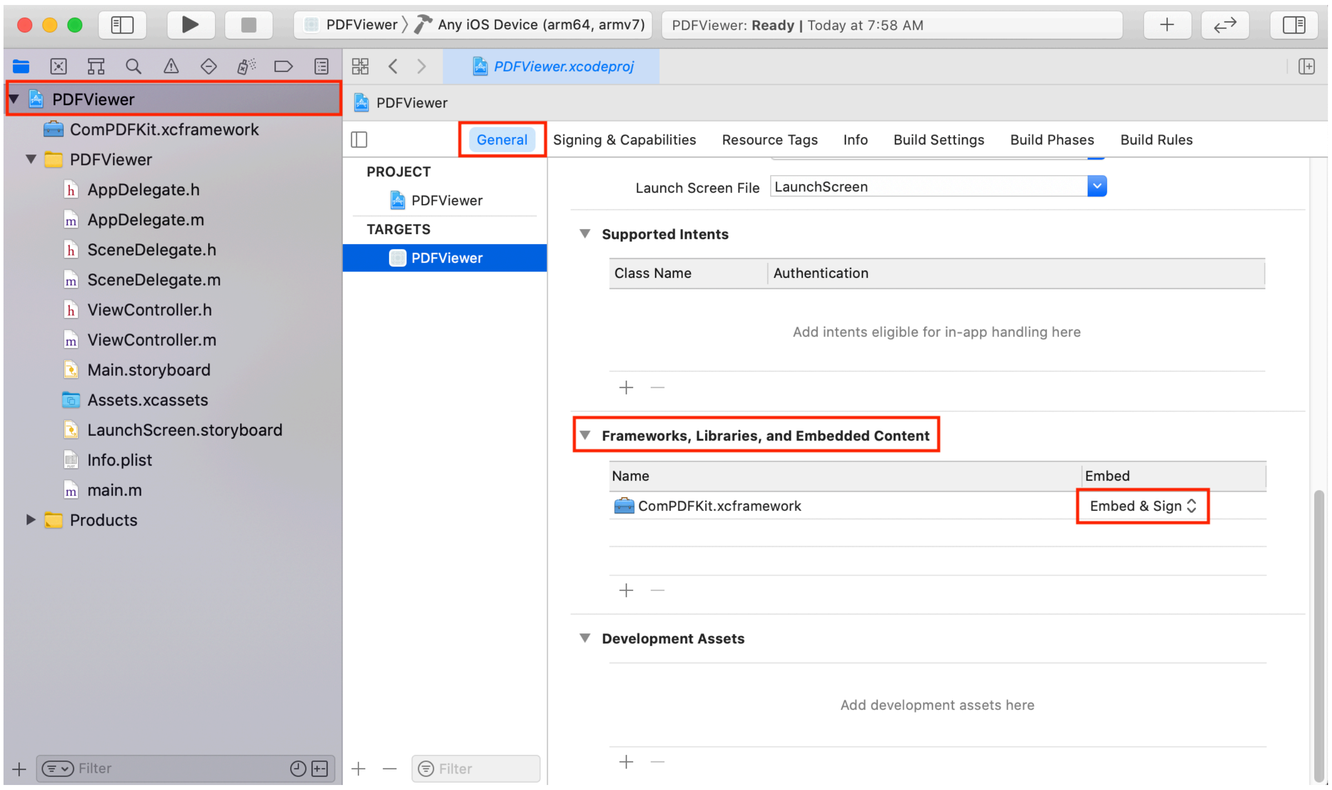
Display Name: PDFViewer
Bundle Identifier: com.PDFViewer
Version: 1.0
Build: 1

Deployment Info

iOS 10.0: ✓ iPhone, ✓ iPad, Mac
Main Interface: Main
Device Orientation: ✓ Portrait, Upside Down, ✓ Landscape Left, ✓ Landscape Right
Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-6

3. Add the dynamic xcframework ***"ComPDFKit.xcframework"*** to the Xcode's **Embedded Binaries**. Left-click the project, find **Embedded Binaries** in the **General** tab, and choose ***"Embed & Sign"*** as shown in Figure 2-7.



Launch Screen File: LaunchScreen

Supported Intents

Class Name	Authentication

Add intents eligible for in-app handling here

Frameworks, Libraries, and Embedded Content

Name	Embed
ComPDFKit.xcframework	Embed & Sign

Development Assets

Add development assets here

Figure 2-7

2.4.3 Apply the License Key

It is important that you set the license key before using any ComPDFKit PDF SDK classes.

```
#import <ComPDFKit/ComPDFKit.h>

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
  // Set your license key here. ComPDFKit is commercial software.
  // Each ComPDFKit license is bound to a specific app bundle id.
  // com.compdfkit.pdfviewer

    [CPDFKit setLicenseKey:@"YOUR_LICENSE_KEY_GOES_HERE"
                    secret:@"YOUR_LICENSE_SECRET_GOES_HERE"];

    return YES;
}
```

2.4.4 Display a PDF Document

So far, we have added ***"ComPDFKit.xcframework"*** to the ***"PDFViewer"*** project, and finished the initialization of the ComPDFKit PDF SDK. Now, let's start building a simple PDF viewer with just a few lines of code.

Then, add the following code to ViewController.m to display a PDF document. It's really easy to present a PDF on screen. All you need is to create a `CPDFDocument` object and then show it with a `CPDFView` object.

```
#import <ComPDFKit/ComPDFKit.h>

- (void)viewDidLoad {
    [super viewDidLoad];

    // Get the path of a PDF
    NSString *pdfPath = @"...";

    // Initialize a CPDFDocument object with the path to the PDF file
    NSURL *url = [NSURL fileURLWithPath:pdfPath];
    CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
    if (document.error && document.error.code != CPDFDocumentPasswordError) {
        return;
    }

    // Initialize a CPDFView object with the size of the entire screen
    CPDFView *pdfView = [[[CPDFView alloc] initWithFrame:self.view.bounds]
autorelease];

    // Set the document to display
    pdfView.document = document;
    // Add the pdfView to the root view
    [self.view addSubview:pdfView];
}
```

2.5 How to Make an iOS App in Objective-C with Default UI

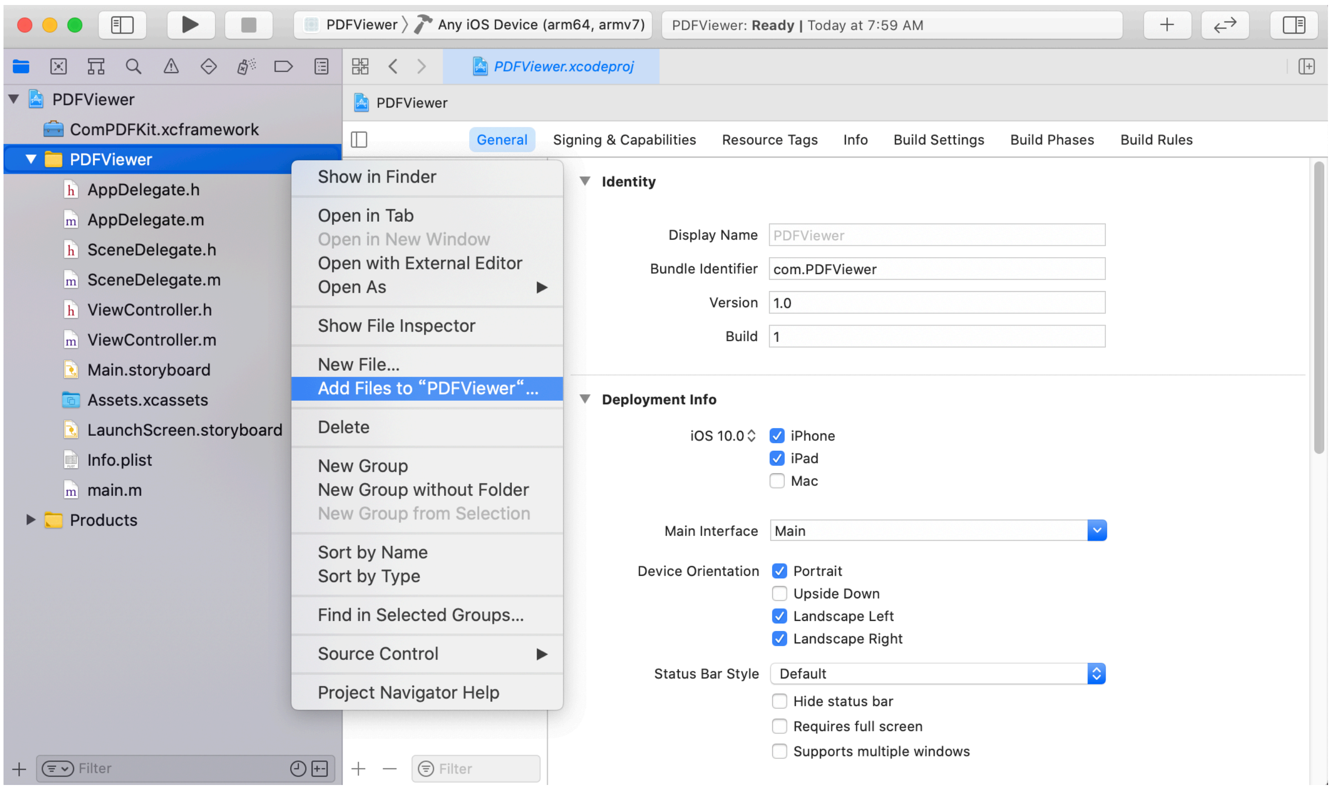
"PDFViewer" for iOS sample projects comes with a default UI design, including the basic UI for the app and the feature modules UI, which are implemented using ComPDFKit PDF SDK and are shipped in the ***" PDFViewer / Source "*** folder. Also included is a `PDFViewController` view controller that contains ready-to-use UI module implementations.

Folder	Description
PDFView	`PDFListView` is a subclass of `CPDFView` that can add, drag and delete annotations, and perform custom drawing on top of the PDF page.
PDFViewerMode	Allows user to pick display modes, switch between different themes and set crop mode.
PDFSlider	Allows user to quickly skip through pages.
PDFBOTA	The `PDFBOTAViewController` class is a container view controller that shows the bookmarks, outline, thumbtail and annotation list controls. A segmented control is used to select which child view controller to display.
PDFSearch	Allows user to search, highlight all instances of a search term, and navigate among search results.
PDFToolbar	`PDFToolbar` is a subclass of `UIView` that can configure the annotations toolbar.
PDFColor	Allows user to customize color and opacity set.
PDFFreehand	The `PDFFreehandViewController` class allows user to change the different properties of an ink annotation. Users can use it to customize selected annotations by changing their various appearance properties.
PDFShape	The `PDFShapeViewController` class allows user to change the different properties of square, circle, and line annotation. Users can use it to customize selected annotations by changing their various appearance properties.
PDFLink	The `PDFLinkViewController` class allows user to change the different properties of a link annotation. Users can use it to customize selected annotations by changing their various appearance properties.
PDFNote	The `PDFNoteViewController` class allows user to read and edit content of note annotation.
PDFStamp	Allows user to create standard stamp, and custom stamp to change text and image.
PDFSignature	Allows user to add electronic drawn signatures to PDF documents.
PDFKeyboardToolbar	The `PDFKeyboardToolbar` class shows freetext annotation properties in a top keyboard view, and users can change the color, font, and opacity with it.
PDFPageEdit	Enables a whole host of document editing features, which includes new page creation, page reordering, rotation, extraction, and deletion.

2.5.1 Integrate Default UI into Your Apps

To add default UI into the ***"PDFViewer"*** project, please follows the steps below:

1. Right-click the ***"PDFViewer"*** folder, select **Add Files to "PDFViewer"...** as shown in Figure 2-8.



Identity

Display Name: PDFViewer
Bundle Identifier: com.PDFViewer
Version: 1.0
Build: 1

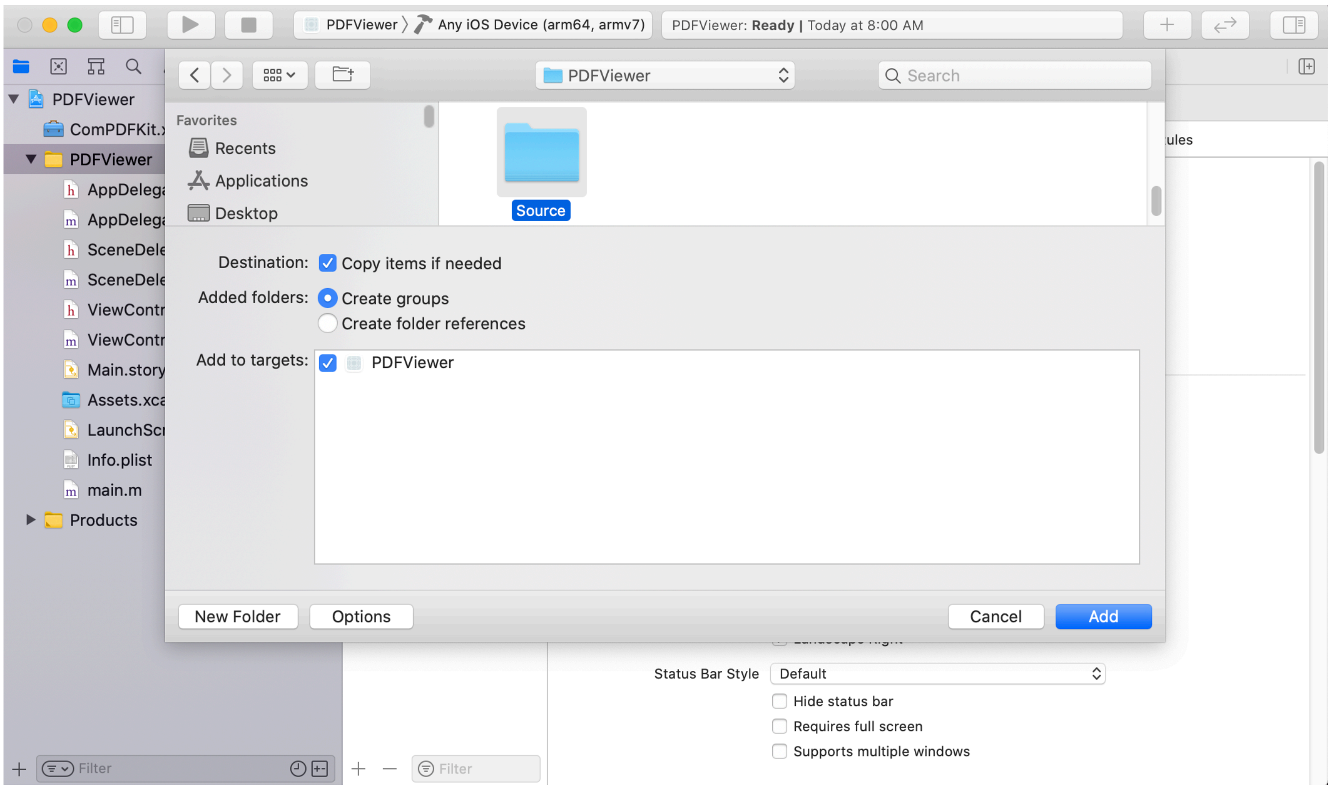
Deployment Info

iOS 10.0: [✓ iPhone] [✓ iPad] Mac
Main Interface: Main
Device Orientation: [✓ Portrait] Upside Down [✓ Landscape Left] [✓ Landscape Right]
Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-8

2. Find and choose ***"PDFViewController.h"***, ***"PDFViewController.m"*** and ***"Source"*** folder in the download package, and then click **Add** as shown in Figure 2-9.

   Note: *Make sure to check the* ***"Copy items if needed"*** *option.*

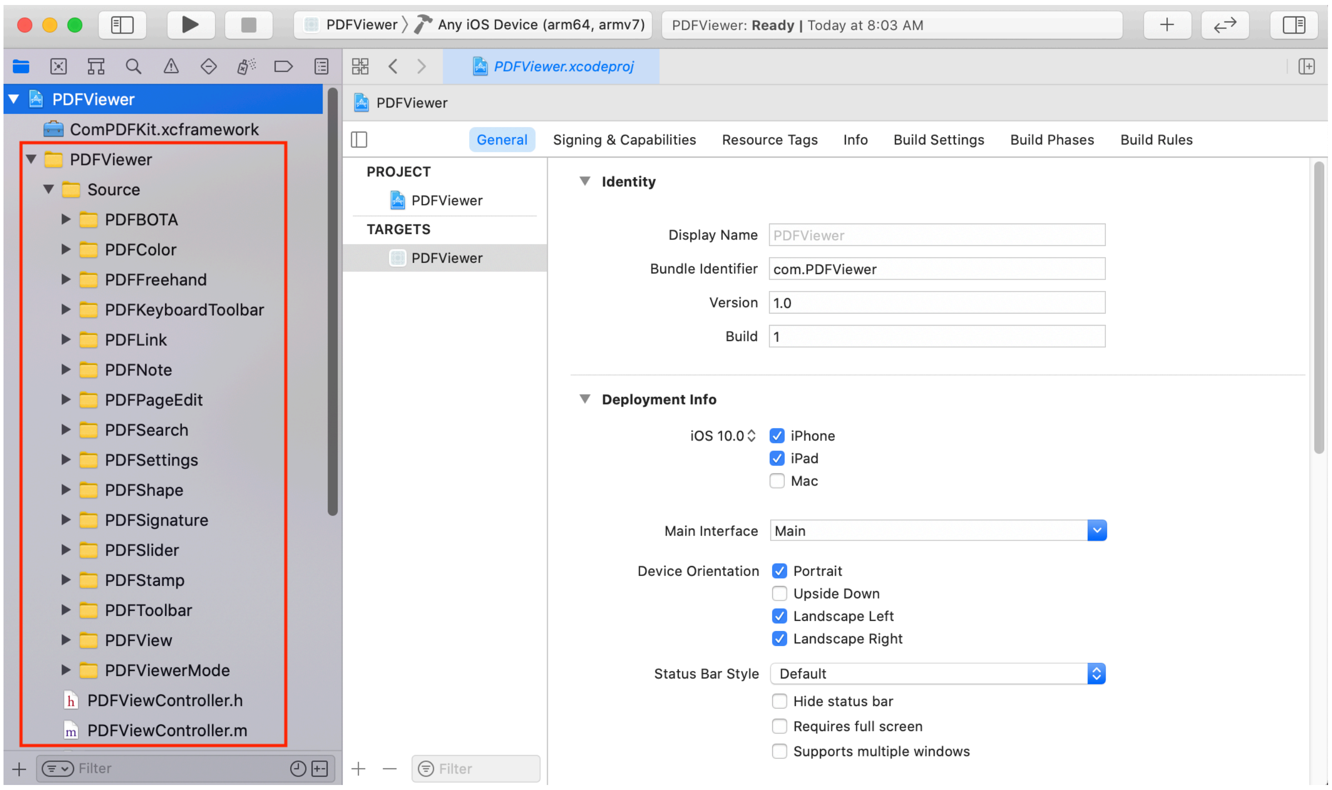


Destination: [✓ Copy items if needed]
Added folders: [✓ Create groups] Create folder references
Add to targets: [✓ PDFViewer]

Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-9

Then, the ***"PDFViewer"*** project will look like the Figure 2-10.



Identity

Display Name: PDFViewer
Bundle Identifier: com.PDFViewer
Version: 1.0
Build: 1

Deployment Info

iOS 10.0: ✓ iPhone, ✓ iPad, Mac
Main Interface: Main
Device Orientation: ✓ Portrait, Upside Down, ✓ Landscape Left, ✓ Landscape Right
Status Bar Style: Default
Hide status bar
Requires full screen
Supports multiple windows

Figure 2-10

3. To protect user privacy, an iOS app linked on or after iOS 10.0, and that accesses the device's privacy-sensitive data, you need to do the following configuration in the ***"Info.plist"***.

```
<key>NSCameraUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSMicrophoneUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSPhotoLibraryAddUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSPhotoLibraryUsageDescription</key>
<string>Your consent is required before you could access the function.</string>
```

2.5.2 How to Initialize the PDFViewController Class

To initialize `PDFViewController` Class, refer to the following method in the `PDFViewController` class.

```
PDFViewController *vc = [[[PDFViewController alloc] initWithFilePath:filePath]
autorelease];
```

2.5.3 How to Handle PDF Document Loading

To handle PDF document loading, refer to the following method in the ***"PDFViewController.m"*** file.

```
- (void)loadDocumentWithFilePath:(NSString *)filePath completion:(void (^)(BOOL
result))completion;
```

2.5.4 How to Use the PDFListView Class

`PDFListView` is a subclass of `CPDFView` that contains operations for annotations.

1. Add support for Annotations

   Set the `annotationMode` property to enter a different annotation mode, and click or drag to add a different annotation in the PDF view. Refer to the following method in the ***"PDFViewController.m"*** file.

   ```
   - (PDFToolbar *)annotationToolbar;
   ```

2. Implementing delegate method

   About Implementing delegate method, refer to the following method in the ***"PDFViewController.m"*** file.

   ```
   // Add and modify text annotation
   - (void)PDFViewPerformOpenNote:(PDFListView *)pdfView forAnnotation:(CPDFAnnotation
   *)annotation;

   // Modify annotation color
   - (void)PDFViewPerformChangeColor:(PDFListView *)pdfView forAnnotation:
   (CPDFAnnotation *)annotation;

   // Share markup annotation
   - (void)PDFViewPerformShare:(PDFListView *)pdfView forAnnotation:
   (CPDFMarkupAnnotation *)annotation;

   // Save stamp image
   - (void)PDFViewPerformSave:(PDFListView *)pdfView forAnnotation:
   (CPDFStampAnnotation *)annotation;

   // Add popup for markup annotation
   - (void)PDFViewPerformPopup:(PDFListView *)pdfView forAnnotation:
   (CPDFMarkupAnnotation *)annotation;

   // Add and modify link annotation
   - (void)PDFViewPerformEditLink:(PDFListView *)pdfView forAnnotation:
   (CPDFLinkAnnotation *)annotation;

   // Signature widget add signature
   ```

```
- (void)PDFViewPerformSignatureWidget:(PDFListView *)pdfView forAnnotation:
(CPDFSignatureWidgetAnnotation *)annotation;

// Share selection text
- (void)PDFViewPerformShare:(PDFListView *)pdfView forSelection:(CPDFSelection
*)selection;

- (void)PDFViewPerformDefine:(PDFListView *)pdfView forSelection:(CPDFSelection
*)selection;

// Search selection text for Google
- (void)PDFViewPerformGoogleSearch:(PDFListView *)pdfView forSelection:
(CPDFSelection *)selection;

// Search selection text for Wiki
- (void)PDFViewPerformWikiSearch:(PDFListView *)pdfView forSelection:(CPDFSelection
*)selection;

// Add Signture annotation
- (void)PDFViewPerformAddSignture:(PDFListView *)pdfView atPoint:(CGPoint)point
forPage:(CPDFPage *)page;

// Add Stamp annotation
- (void)PDFViewPerformAddStamp:(PDFListView *)pdfView atPoint:(CGPoint)point
forPage:(CPDFPage *)page;

// Add Image annotation
- (void)PDFViewPerformAddImage:(PDFListView *)pdfView atPoint:(CGPoint)point
forPage:(CPDFPage *)page;

- (void)PDFViewPerformTouchEnded:(PDFListView *)pdfView;

// Page jump
- (void)PDFViewPerformWillGoTo:(PDFListView *)pdfView pageIndex:
(NSInteger)pageIndex;
```

2.5.5 How to Use the PDFKeyboardToolbar Class

PDFKeyboardToolbar is the keyboard toolbar of the Freetext annotation input box.

1. Initialize the PDFKeyboardToolbar class

Initialize the `PDFKeyboardToolbar` in the CPDFView delegate method, and refer to the following method in the ***"PDFViewController.m"*** file.

```
- (void)PDFViewShouldBeginEditing:(CPDFView *)pdfView textView:(UITextView
*)textView forAnnotation:(CPDFFreeTextAnnotation *)annotation {
    self.annotationFreeText = annotation;
    PDFKeyboardToolbar *keyboardToolbar = [[[PDFKeyboardToolbar alloc]
initWithFontName:annotation.font.fontName fontSize:annotation.font.pointSize]
autorelease];
    keyboardToolbar.delegate = self;
    [keyboardToolbar refreshFontName:annotation.font.fontName
fontSize:annotation.font.pointSize opacity:annotation.opacity];
    [keyboardToolbar bindToTextView:textView];
}
```

2. Implementing delegate method

   About Implementing delegate method, refer to the following method in the ***"PDFViewController.m"*** file.

```
// keyboard should dissmiss
- (void)keyboardShouldDissmiss:(PDFKeyboardToolbar *)toolbar;

// Modify font name of freetext annotation
- (void)keyboard:(PDFKeyboardToolbar *)toolbar updateFontName:(NSString *)fontName;

// Modify font size of freetext annotation
- (void)keyboard:(PDFKeyboardToolbar *)toolbar updateFontSize:(CGFloat)fontSize;

// Modify text color of freetext annotation
- (void)keyboard:(PDFKeyboardToolbar *)toolbar updateTextColor:(UIColor
*)textColor;

// Modify text opacity of freetext annotation
- (void)keyboard:(PDFKeyboardToolbar *)toolbar updateOpacity:(CGFloat)opacity;
```

2.6 ARC Compatibility

ComPDFKit PDF SDK requires non-ARC. If you wish to use ComPDFKit PDF SDK in a ARC project, just add the -fno-objc-arc compiler flag. To do this, go to the Build Phases tab in your target settings, open the Compile Sources group, and double-click and type -fno-objc-arc into the popover.

2.7 Swift Compatibility

To use the ComPDFKit Objective-C Framework in your Swift project, you have to create a Swift Bridging Header file in that project. The best way is to create the .h file Manually.

First, add a header file to your project with the name: MyProjectName-Bridging-Header.h. This will be the single header file where you import any Objective C code you need your Swift code to have access.

Find Swift Compiler - Code Generation section in your project build settings. Add the path to your bridging header file next to Objective C Bridging Header from the project root folder. It should be MyProject/MyProject-Bridging-Header.h

How to use iOS Objective C Framework in Swift project?

3 Guides

If you're interested in all of the features mentioned in Overview section, please go through our guides to quickly add PDF viewing, annotating, and editing to your application. The following sections list some examples to show you how to add document functionalities to iOS apps using our Swift and Objective-C APIs.

3.1 Basic Operations

There are a few commonly used basic operations when working with documents.

3.1.1 Open a Document

- Open a Local File

```
// Get the path of a PDF
NSString *pdfPath = @"...";

// Initialize a CPDFDocument object with the path to the PDF file
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
if (document.error && document.error.code != CPDFDocumentPasswordError) {
    return;
}
```

- Create a New File

```
CPDFDocument *document = [[[CPDFDocument alloc] init] autorelease];
```

3.1.2 Save a Document

To save a PDF document to path use `CPDFDocument::writeToURL:`.

If this path is the original path, the PDF document will be saved incrementally. Use the incremental mode if you are concerned about saving time. If you use this mode, any changes to the document, even deleting annotations, will result in appending to the PDF file.

3.2 Viewer

The viewer provides fast and battle-tested rendering engine with a wide range of advanced features including display modes, content selecting, zoom level setting and page navigation, which offers developers a way to quickly embed a highly configurable PDF viewer in any iOS application.

3.2.1 Display Modes

- Scroll Direction

  When scrolling through pages of a document in `CPDFView`, the scrolling direction can be changed by setting the page display direction.

  - Vertical scrolling mode (also known as continuous mode) can be enabled by setting the page display direction to `CPDFDisplayDirectionVertical`.

    ```
    pdfView.displayDirection = CPDFDisplayDirectionVertical;
    ```

  - Horizontal scrolling mode (also known as non-continuous mode) can be enabled by setting the page display direction to `CPDFDisplayDirectionHorizontal`.

    ```
    pdfView.displayDirection = CPDFDisplayDirectionHorizontal;
    ```

  You also can configure the scroll direction of `CPDFView` in `CPDFKitConfig`.

  ```
  CPDFKitShareConfig.displayDirection = CPDFDisplayDirectionVertical;
  ```

- Two-up Mode

  Displays two pages side-by-side.

  ```
  pdfView.displayTwoUp = YES;
  ```

- Cover Mode

  Show cover page during two-up.

  ```
  pdfView.displayTwoUp = YES;
  pdfView.displaysAsBook = YES;
  ```

- Crop Mode

  Automatically trim PDF files white margins to resize pages. Crop mode can be enabled by setting the page display crop to `Yes`.

```
pdfView.displayCrop = YES;
```

Note: *You must call `layoutDocumentView` method explicitly if using these `CPDFView` properties (`displayDirection`, `displaysPageBreaks`, `pageBreakMargins`, `displayTwoUp`, `displaysAsBook`, `displayCrop`).*

3.2.2 PDF Navigation

- Page Navigation

  After loading a PDF document, you can programmatically interact with it, which allows you to scroll to different pages or destinations. All of the interaction APIs are available on `CPDFView`.

  - Scrolls to the specified page, use function `CPDFView::goToPageIndex:animated:`.
  - Goes to the specified destination, destinations include a page and a point on the page specified in page space, use function `CPDFView::goToDestination:animated:`.
  - Goes to the specified rectangle on the specified page, use function `CPDFView::goToRect:onPage:animated:`.

    This allows you to scroll the `CPDFView` object to a specific `CPDFAnnotation` or `CPDFSelection` object, because both of these objects have bounds methods that return an annotation or selection position in page space.
    Note: *This method's rect is specified in page-space coordinates. Page space is a coordinate system with the origin at the lower-left corner of the current page.*

- Outline

  Outline allows users to quickly locate and link their point of interest within a PDF document. Each outline contains a destination or actions to describe where it links to. It is a tree-structured hierarchy, so function `CPDFDocument::outlineRoot` must be called first to get the root of the whole outline tree before accessing the outline tree. Here, "root outline" is an abstract object which can only have some child outline without the next sibling outline and any data (includes outline data, destination data, and action data). It cannot be shown on the application UI since it has no data. You can also use function `CPDFDocument::setNewOutlineRoot` to create a new root outline.

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

  - To access the parent outline, use function `CPDFOutline::parent`.
  - To access the child outline.

```
- (NSArray<CPDFOutline *> *)childOutline:(CPDFOutline *)outline {
    NSUInteger numberOfChildren = [outline numberOfChildren];
    NSMutableArray *child = [NSMutableArray array];
    for (int i=0; i<numberOfChildren; i++) {
        [child addObject:[outline childAtIndex:i]];
    }
    return child;
}
```

  - To insert a new outline, use function `CPDFOutline::insertChildAtIndex:`.
  - To remove an outline, use function `CPDFOutline::removeFromParent`.
  - To move an outline, use function `CPDFOutline::insertChild:atIndex:`. When moving items around within an outline hierarchy, you should retain the item and call `CPDFOutline::removeFromParent` first.

- Bookmarks

  Since each bookmark is associated with a specific page, it provides the ability to link to a different page in a document allowing the user to navigate interactively from one part of the document to another.

  - To access bookmarks, use function `CPDFDocument::bookmarks`.
  - To access a bookmark for page, use function `CPDFDocument::bookmarkForPageIndex:`.
  - To add a new bookmark, use function `CPDFDocument::addBookmark:forPageIndex:`.
  - To remove a bookmark, use function `CPDFDocument::removeBookmarkForPageIndex:`.

3.2.3 Text Search & Selection

- Text Search

  ComPDFKit PDF SDK offers developers an API for programmatic full-text search, as well as UI for searching and highlighting relevant matches.

  - Asynchronously finds all instances of the specified string in the document, use function `CPDFDocument::beginFindString:withOptions:`, which returns immediately. It causes delegate methods to be called when searching begins and ends, on each search hit, and when the search proceeds to a new page.

```
/**
 * Called when the beginFindString:withOptions: or findString:withOptions:
method begins finding.
 */
- (void)documentDidBeginDocumentFind:(CPDFDocument *)document;
/**
 * Called when the beginFindString:withOptions: or findString:withOptions:
method returns.
 */
- (void)documentDidEndDocumentFind:(CPDFDocument *)document;
/**
```

```
 * Called when a find operation begins working on a new page of a document.
 */
- (void)documentDidBeginPageFind:(CPDFDocument *)document pageAtIndex:
(NSUInteger)index;
/**
 * Called when a find operation finishes working on a page in a document.
 */
- (void)documentDidEndPageFind:(CPDFDocument *)document pageAtIndex:
(NSUInteger)index;
/**
 * Called when a string match is found in a document.
 *
 * @discussion To determine the string selection found, use the selection.
 */
- (void)documentDidFindMatch:(CPDFSelection *)selection;
```

- Synchronously finds all instances of the specified string in the document, use function `CPDFDocument::findString:withOptions:`. Each hit gets added to an `NSArray` object as a `CPDFSelection` object. If there are no hits, this method returns an empty array. Use this method when the complete search process will be brief and when you don't need the flexibility or control offered by `CPDFDocument::beginFindString:withOptions:`.
- Cancels a search initiated with `CPDFDocument::beginFindString:withOptions:`, use function `CPDFDocument::cancelFindString`.
- Highlighting Search Results, `CPDFView` offers a way to both add and clear search results, use function `CPDFView::setHighlightedSelection:animated:`.

- Text Selection

  PDF text contents are stored in `CPDFPage` objects which are related to a specific page. `CPDFPage` 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 bounds and more.

  How to get the text bounds on a page by selection:

```
- (CPDFSelection *)selectionForPage:(CPDFPage *)page fromPoint:(CGPoint)fPoint
toPoint:(CGPoint)tPoint {
    NSInteger fCharacterIndex = [page characterIndexAtPoint:fPoint];
    NSInteger tCharacterIndex = [page characterIndexAtPoint:tPoint];
    NSRange range = NSMakeRange(fCharacterIndex, tCharacterIndex - fCharacterIndex
+ 1);
    CPDFSelection *selection = [page selectionForRange:range];
    return selection;
}
```

3.2.4 Zooming

ComPDFKit PDF SDK provides super zoom out and in to unlock more zoom levels, and pinch-to-zoom or double tap on the specific area to perform a smart page content analysis, or you can programmatically interact with it by using the following method.

- Manual Zooming

  You can use `CPDFView::setScaleFactor:animated:` to zoom the current document.

- Disabling Zooming

  Zooming can be disabled by setting the `scrollEnabled` to `NO` on `CPDFView`.

  ```
  pdfView.scrollEnabled = NO;
  ```

3.2.5 Themes

`CPDFView` has four special color modes: dark mode, sepia mode, reseda mode, and custom color mode.

In dark mode, colors are adjusted to improve reading at night or in a poorly-lit environment, in sepia mode, background color is set to emulate the look of an old book, in reseda mode, light-green background is displayed to protect your eyes after long-time reading, and in custom color mode, you can set a custom color for the background color.

Note: *Changing the appearance mode will change the PDF rendering style, but it does not modify the PDF on disk.*

To set the color mode:

1. Find the constant value of the color mode

Themes	Constant value
Normal color mode	`CPDFDisplayModeNormal`
Dark mode	`CPDFDisplayModeNight`
Sepia mode	`CPDFDisplayModeSoft`
Reseda mode	`CPDFDisplayModeGreen`
Custom color mode	`CPDFDisplayModeCustom`

2. Call `CPDFView::setDisplayMode:`.
3. If you are using `CPDFDisplayModeCustom`, call `CPDFView::setDisplayModeCustomColor:` to set the background color.
4. Update `CPDFView` to redraw the contents.

   ```
   [pdfView layoutDocumentView];
   ```

You also can configure the color mode of `CPDFView` in `CPDFKitConfig`.

```
CPDFKitShareConfig.displayMode = CPDFDisplayModeCustom;
CPDFKitShareConfig.displayModeCustomColor = [UIColor whiteColor];
```

3.2.6 Text Reflow

Rearrange text to fit the device screen size for displaying the same layout by using the following method.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

NSRange range = NSMakeRange(0, page.numberOfCharacters);
NSString *string = [page stringForRange:range];
```

3.2.7 Rendering

The `CPDFView` class calls `drawPage:toContext:` as necessary for each visible page that requires rendering. You can override this method to draw on top of a PDF page. In this case, invoke this method on `super` and then perform custom drawing on top of the PDF page. Do not invoke this method, except by invoking it on `super` from a subclass.

3.3 Annotations

In addition to its primary textual content, a PDF file can contain annotations that represent links, form elements, highlighting circles, textual notes, and so on. Each annotation is associated with a specific location on a page and may offer interactivity with the user. Annotations allow users to mark up and comment on PDFs without altering the original author's content.

ComPDFKit PDF SDK supports most annotation types defined in PDF Reference and provides APIs for annotation creation, properties access and modification, appearance setting, and drawing.

3.3.1 Annotation Types

ComPDFKit PDF SDK supports all common annotation types:

- Note
- Link
- Free Text
- Shapes: Square, Circle, and Line

- Markup: Highlight, Underline, Strikeout, and Squiggly
- Stamp
- Ink
- Sound

These are standard annotations (as defined in the PDF Reference) that can be read and written by many apps, such as Adobe Acrobat and Apple Preview.

3.3.2 Access Annotations

`CPDFAnnotation` is the base class for all annotations. A `CPDFAnnotation` object by itself is not useful, only subclasses (like `CPDFCircleAnnotation`, `CPDFTextAnnotation`) are interesting. In parsing a PDF however, any unknown or unsupported annotations will be represented as this base class.

To access the list of annotations by using the following method:

```
- (NSArray<CPDFAnnotation *>)annotationsWithDocument:(CPDFDocument *)document {
    NSMutableArray *annotations = [NSMutableArray array];
    for (int i=0; i<document.pageCount; i++) {
        CPDFPage *page = [document pageAtIndex:i];
        [annotations addObjectsFromArray:[page annotations]];
    }
}
```

The elements of the array will most likely be typed to subclasses of the `CPDFAnnotation` class.

3.3.3 Create & Edit Annotations

ComPDFKit PDF SDK includes a wide variety of standard annotations, and each of them is added to the project in a similar way.

- Note

  To add a sticky note (text annotation) to a PDF Document page by using the following method.

  ```
  NSURL *url = [NSURL fileURLWithPath:pdfPath];
  CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
  CPDFPage *page = [document pageAtIndex:0];

  CPDFTextAnnotation *text = [[[CPDFTextAnnotation alloc] initWithDocument:document]
  autorelease];
  text.contents = @"test";
  text.bounds = CGRectMake(0, 0, 50, 50);
  text.color = [UIColor yellowColor];
  [page addAnnotation:text];
  ```

- Link

To add a hyperlink or intra-document link annotation to a PDF Document page by using the following method.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

CPDFDestination *dest = [[[CPDFDestination alloc] initWithDocument:document
pageIndex:1] autorelease];
CPDFLinkAnnotation *link = [[[CPDFLinkAnnotation alloc] initWithDocument:document]
autorelease];
link.bounds = CGRectMake(0, 0, 50, 50);
link.destination = dest;
//link.URL = @"https://www.";
[page addAnnotation:link];
```

- Free Text

  To add a free text annotation to a PDF Document page by using the following method.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

CPDFFreeTextAnnotation *freeText = [[[CPDFFreeTextAnnotation alloc]
initWithDocument:document] autorelease];
freeText.contents = @"test";
freeText.bounds = CGRectMake(0, 0, 50, 50);
freeText.font = [UIFont systemFontOfSize:12];
freeText.fontColor = [UIColor redColor];
freeText.alignment = NSTextAlignmentLeft;
[page addAnnotation:freeText];
```

- Shapes

  To add a shape annotation to a PDF Document page by using the following method.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

CPDFBorder *border = [[[CPDFBorder alloc] initWithStyle:CPDFBorderStyleDashed
                                                lineWidth:1
                                              dashPattern:@[@(2), @(1)]]
autorelease];

// Square
CPDFSquareAnnotation *square = [[[CPDFSquareAnnotation alloc]
initWithDocument:document] autorelease];
```

```
square.bounds = CGRectMake(0, 0, 50, 50);
square.color = [UIColor redColor];
square.interiorColor = [UIColor yellowColor];
square.opacity = 0.5;
square.interiorOpacity = 0.5;
square.border = border;
[page addAnnotation:square];

// Circle
CPDFCircleAnnotation *circle = [[[CPDFCircleAnnotation alloc]
initWithDocument:document] autorelease];
circle.bounds = CGRectMake(0, 0, 50, 50);
circle.color = [UIColor redColor];
circle.interiorColor = [UIColor yellowColor];
circle.opacity = 0.5;
circle.interiorOpacity = 0.5;
circle.border = border;
[page addAnnotation:circle];

// Line
CPDFLineAnnotation *line = [[[CPDFLineAnnotation alloc] initWithDocument:document]
autorelease];
line.startPoint = CGPointMake(0, 0);
line.endPoint = CGPointMake(50, 50);
line.startLineStyle = CPDFLineStyleNone;
line.endLineStyle = CPDFLineStyleClosedArrow;
line.color = [UIColor redColor];
line.interiorColor = [UIColor yellowColor];
line.opacity = 0.5;
line.interiorOpacity = 0.5;
line.border = border;
[page addAnnotation:line];
```

Note: *`CPDFLineAnnotation` properties (`startPoint`, `endPoint`) point is specified in page-space coordinates. Page space is a coordinate system with the origin at the lower-left corner of the current page.*

- Markup

  To add a highlight annotation to a PDF Document page by using the following method, and add other markup annotations in a similar way.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

CPDFSelection *selection = ...;
NSMutableArray *quadrilateralPoints = [NSMutableArray array];
for (CPDFSelection *selection in selection.selectionsByLine) {
    CGRect bounds = selection.bounds;
    [quadrilateralPoints addObject:[NSValue
valueWithCGPoint:CGPointMake(CGRectGetMinX(bounds), CGRectGetMaxY(bounds))]];
    [quadrilateralPoints addObject:[NSValue
valueWithCGPoint:CGPointMake(CGRectGetMaxX(bounds), CGRectGetMaxY(bounds))]];
    [quadrilateralPoints addObject:[NSValue
valueWithCGPoint:CGPointMake(CGRectGetMinX(bounds), CGRectGetMinY(bounds))]];
    [quadrilateralPoints addObject:[NSValue
valueWithCGPoint:CGPointMake(CGRectGetMaxX(bounds), CGRectGetMinY(bounds))]];
}

CPDFMarkupAnnotation *highlight = [[[CPDFMarkupAnnotation alloc]
initWithDocument:document markupType:CPDFMarkupTypeHighlight] autorelease];
highlight.color = [UIColor yellowColor];
highlight.quadrilateralPoints = quadrilateralPoints;
[page addAnnotation:highlight];
```

- Stamp

  To add standard, text, and image stamps to a PDF document page by using the following method.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

// Standard
CPDFStampAnnotation *standard = [[[CPDFStampAnnotation alloc]
initWithDocument:document type:0] autorelease];
[page addAnnotation:standard];

// Text
CPDFStampAnnotation *text = [[[CPDFStampAnnotation alloc] initWithDocument:document
text:@"test" detailText:@"detail text" style:CPDFStampStyleRed
shape:CPDFStampShapeArrowLeft] autorelease];
[page addAnnotation:text];

// Image
CPDFStampAnnotation *image = [[[CPDFStampAnnotation alloc]
initWithDocument:document image:[UIImage imageNamed:@""]] autorelease];
[page addAnnotation:image];
```

3.3.4 Delete Annotations

To remove an annotation from a document.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

CPDFAnnotation *annotation = [[page annotations] objectAtIndex:0];
[page removeAnnotation:annotation];
```

3.3.5 Annotation Appearances

Annotations may contain properties that describe their appearance — such as annotation color or shape. However, these don't guarantee that the annotation will be displayed the same in different PDF viewers. To solve this problem, each annotation can define an appearance stream that should be used for rendering the annotation.

ComPDFKit PDF SDK will update the annotation appearance by default when you modify the annotation properties. You can also manually update the appearance by calling the `updateAppearanceStream` method, but you must call the `updateAppearanceStream` method manually when you modify the bounds of the FreeText, Stamp, Signature annotation, refer to the following method in the `CPDFAnnotation` class.

```
- (void)updateAppearanceStream;
```

It's easy to set up an annotation to show a custom appearance stream. This is typically done with stamp annotations because they have few other properties. A stamp annotation used this way is usually called an image annotation.

The following part introduces how to set annotation appearance that does not match page rotation.

If set, do not rotate the annotation's appearance to match the rotation of the page. The upper-left corner of the annotation bounds shall remain in a fixed location on the page.

```
CPDFKitShareConfig.enableAnnotationNoRotate = YES;
```

In addition, when it comes to the FreeText annotation, refer to the following method in the `PDFListView` class.

```
- (void)addAnnotationFreeTextAtPoint:(CGPoint)point forPage:(CPDFPage *)page;
- (void)drawPage:(CPDFPage *)page toContext:(CGContextRef)context;
- (void)moveAnnotation:(CPDFAnnotation *)annotation fromPoint:(CGPoint)fromPoint
toPoint:(CGPoint)toPoint forType:(PDFAnnotationDraggingType)draggingType;
```

3.3.6 Import & Export Annotations

XFDF is an XML-based standard from Adobe XFDF for encoding annotations. An XFDF file will contain a snapshot of a PDF document's annotations and forms. It's compatible with Adobe Acrobat and several other third-party frameworks. ComPDFKit supports both reading and writing XFDF.

- Importing from XFDF

  You can import annotations and form fields from an XFDF file to a document like so:

  ```
  NSURL *url = [NSURL fileURLWithPath:pdfPath];
  CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

  NSString *path = [NSString stringWithString:xfdfPath];
  [document importAnnotationFromXFDFPath:path];
  ```

- Exporting to XFDF

  You can export annotations and form fields from a document to an XFDF file like so:

  ```
  NSURL *url = [NSURL fileURLWithPath:pdfPath];
  CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

  NSString *path = [NSString stringWithString:xfdfPath];
  [document exportAnnotationToXFDFPath:path];
  ```

3.3.7 Flatten Annotations

Annotation flattening refers to the operation that changes annotations into a static area that is part of the PDF document, just like the other text and images in the document. When flattening an annotation, the annotation is removed from the document, while its visual representation is kept intact. A flattened annotation is visible but is non-editable by your users or by your app.

Annotations in a PDF document can be flattened in the ComPDFKit PDF SDK by saving the document and choosing the Flatten mode.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

NSURL *surl = [NSURL fileURLWithPath:savePath];
[document writeFlattenToURL:surl];
```

3.3.8 Predefine Annotations

ComPDFKit PDF SDK has default values for some annotation properties, such as colors and line widths for ink annotations.

This is all handled in `CPDFKitConfig`, which is a global singleton. You can access it with `CPDFKitShareConfig`.

The current set of defaults is configured on the first run and saved in `NSUserDefaults`.

```
// Author
CPDFKitShareConfig.annotationAuthor = @"";

// Color
CPDFKitShareConfig.highlightAnnotationColor = [UIColor yellowColor];
CPDFKitShareConfig.underlineAnnotationColor = [UIColor blueColor];
CPDFKitShareConfig.strikeoutAnnotationColor = [UIColor redColor];
CPDFKitShareConfig.squigglyAnnotationColor = [UIColor blackColor];
CPDFKitShareConfig.shapeAnnotationColor = [UIColor redColor];
CPDFKitShareConfig.shapeAnnotationInteriorColor = nil;
CPDFKitShareConfig.freehandAnnotationColor = [UIColor redColor];

// Opacity
CPDFKitShareConfig.markupAnnotationOpacity = 0.5;
CPDFKitShareConfig.shapeAnnotationOpacity = 1.0;
CPDFKitShareConfig.shapeAnnotationInteriorOpacity = 0.0;
CPDFKitShareConfig.freehandAnnotationOpacity = 1.0;

// Border Width
CPDFKitShareConfig.shapeAnnotationBorderWidth = 1.0;
CPDFKitShareConfig.freehandAnnotationBorderWidth = 1.0;
```

3.4 Forms

A PDF document may contain any number of form fields that allow a user to enter information on a PDF page. An interactive form (sometimes referred to as an *AcroForm*) is a collection of fields for gathering information interactively from the user. Under the hood, PDF form fields are a type of PDF annotation called widget annotations.

ComPDFKit PDF SDK fully supports reading, filling, creating, and editing PDF forms and provides utility methods to make working with forms simple and efficient.

3.4.1 Supported Form Fields

ComPDFKit PDF SDK supports all form types specified by the PDF specification (such as text boxes, checkboxes, radio buttons, drop-down lists, pushbuttons, and signatures).

`CPDFWidgetAnnotation` is the base class for all form fields, and `CPDFWidgetAnnotation` is subclass for `CPDFAnnotation`. A `CPDFWidgetAnnotation` object by itself is not useful, only subclasses (`CPDFButtonWidgetAnnotation`, `CPDFChoiceWidgetAnnotation`, `CPDFTextWidgetAnnotation`, `CPDFSignatureWidgetAnnotation`) are interesting. In parsing a PDF however, any unknown or unsupported form fields will be represented as this base class.

We have to differentiate between field types and annotation objects:

Type	Annotation Object
Check, Radio, and Push Buttons	`CPDFButtonWidgetAnnotation`
List and Combo Boxes	`CPDFChoiceWidgetAnnotation`
Text	`CPDFTextWidgetAnnotation`
Signatures	`CPDFSignatureWidgetAnnotation`

3.4.2 Create & Edit Form Fields

Create form fields works the same as adding any other annotation, as can be seen in the guides for programmatically creating annotations.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
CPDFPage *page = [document pageAtIndex:0];

NSMutableArray *items = [NSMutableArray array];
CPDFChoiceWidgetItem *item1 = [[[CPDFChoiceWidgetItem alloc] init] autorelease];
item1.value = @"1";
item1.string = @"a";
[items addObject:item1];
CPDFChoiceWidgetItem *item2 = [[[CPDFChoiceWidgetItem alloc] init] autorelease];
item2.value = @"2";
item2.string = @"b";
[items addObject:item2];

CPDFChoiceWidgetAnnotation *widget = [[[CPDFChoiceWidgetAnnotation alloc]
initWithDocument:document listChoice:YES] autorelease];
widget.items = items;
[page addAnnotation:widget];
```

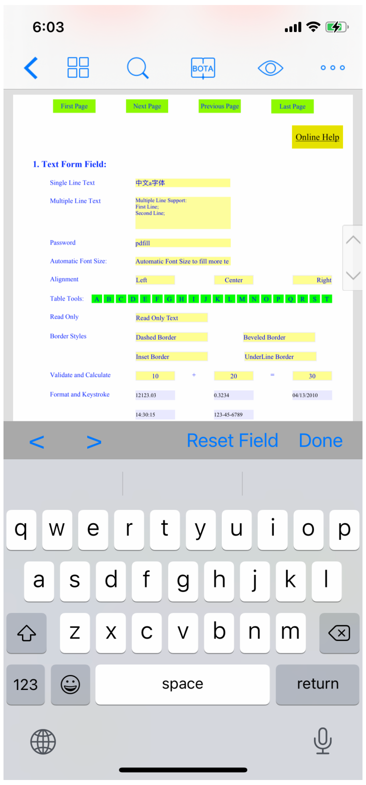
3.4.3 Delete Form Fields

Delete form fields works the same as deleting annotations, and check delete annotations in the guides to see more.

3.4.4 Fill Form Fields

ComPDFKit PDF SDK fully supports the AcroForm standard, and forms can be viewed and filled inside the `CPDFView`.

To fill in a text form element, tap it and then type text using either the onscreen keyboard or an attached hardware keyboard. Then tap either the Done button above the keyboard or any blank area on the page to deselect the form element, which will commit the changes.



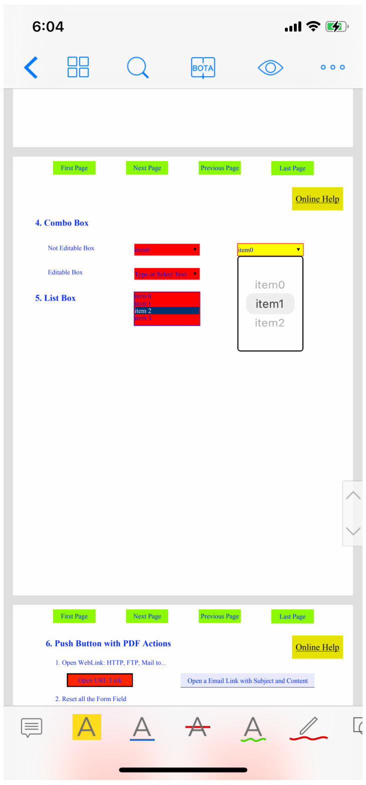
1. Text Form Field:

Single Line Text	中文a字体			

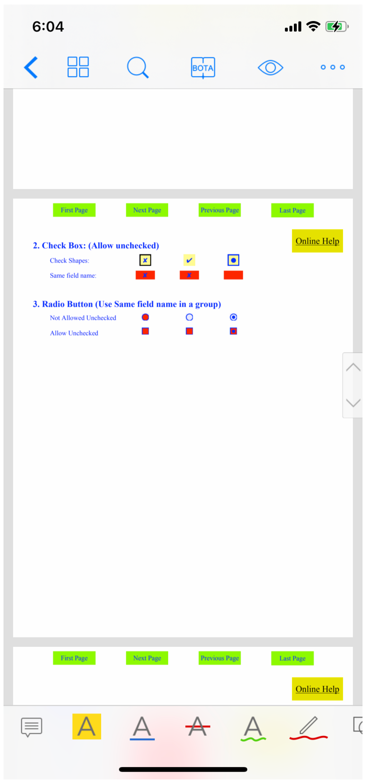
Multiple Line Text	Multiple Line Support:
	First Line;
	Second Line;			

Password	pdfill			
Automatic Font Size:	Automatic Font Size to fill more te			
Alignment	Left		Center	Right
Table Tools:	A B C D E F G H I J K L M N O P Q R S T			
Read Only	Read Only Text			
Border Styles	Dashed Border		Beveled Border	
	Inset Border		UnderLine Border	
Validate and Calculate	10	+	20	= 30
Format and Keystroke	12123.03		0.3234	04/13/2010
	14:30:15		123-45-6789	

To set the value of a choice form element (a list or combo box), tap the element, and then select an item from the list, or type in a custom item.



To enable or disable a checkbox form element, tap it to toggle its state. And you can set the selection of a radio button form element by tapping the desired item.



While a form element is selected (focused), the left and right arrows above the keyboard may be used to move the focus sequentially between all the form elements on the page.

The following example demonstrates how form fields can be queried and filled with code:

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
```

```
CPDFPage *page = [document pageAtIndex:0];

NSArray *annotations = [page annotations];
for (CPDFAnnotation *annotation in annotations) {
    if ([annotation isKindOfClass:[CPDFTextWidgetAnnotation class]]) {
        [(CPDFTextWidgetAnnotation *)annotation setStringValue:@""];
    } else if ([annotation isKindOfClass:[CPDFButtonWidgetAnnotation class]]) {
        if (CPDFWidgetRadioButtonControl == [(CPDFButtonWidgetAnnotation *)annotation
controlType]) {
            [(CPDFButtonWidgetAnnotation *)annotation setState:1];
        }
    } else if ([annotation isKindOfClass:[CPDFChoiceWidgetAnnotation class]]) {
        [(CPDFChoiceWidgetAnnotation *)annotation setSelectItemAtIndex:0];
    }
}
```

3.4.5 Flatten PDF Forms

PDF Form flattening works the same as annotation flattening, and refer to annotation flattening in the guides to see more.

3.5 Document Editor

ComPDFKit provides a wide range of APIs for document editing operations. These are mostly available through the `CPDFDocument` and `CPDFPage` classes.

ComPDFKit benefits include:

- PDF Manipulation
    - Split pages
    - Merge pages
    - Extract pages
- Page Edit
    - Delete pages
    - Insert pages (choose from another document, a blank page, or an image)
    - Move pages
    - Rotate pages
    - Exchange pages
    - Replace pages
    - Crop pages
- Edit Document Information
- Extract Images

3.5.1 PDF Manipulation

- Split Pages

  `CPDFDocument` can extract ranges of pages from one document and put them into another document. If you run this operation multiple times with different page indexes, you can effectively split a PDF into as many documents as you require.

  To split a PDF document into multiple pages, please use the following method:

  1. Create a blank PDF document.

     ```
     CPDFDocument *document = [[[CPDFDocument alloc] init] autorelease];
     ```

  2. Open a PDF document that contains the pages you want to split.

     ```
     // File path
     NSString *path1 = @"...";
     NSURL *url1 = [NSURL fileURLWithPath:path1];
     CPDFDocument *document1 = [[[CPDFDocument alloc] initWithURL:url1]
     autorelease];
     ```

  3. Extract specific pages from the PDF document that you just opened, and import them into the blank PDF document.

     ```
     // Pages that need to be split, e.g. 2 to 5 pages
     NSIndexSet *indexSet = [NSIndexSet indexSetWithIndexesInRange:NSMakeRange(1,
     4)];
     [document importPages:indexSet fromDocument:document1 atIndex:0];
     ```

  4. Save the document.

     ```
     // Save path
     NSString *path = @"...";
     NSURL *url = [NSURL fileURLWithPath:path];
     [document writeToURL:url];
     ```

- Merge Pages

  ComPDFKit allows you to instantiate multiple `CPDFDocument`, and you can use the `CPDFDocument` API to merge multiple PDF files into a single one.

  To merge PDF documents into one file, please use the following method:

  1. Create a blank PDF document.

     ```
     CPDFDocument *document = [[[CPDFDocument alloc] init] autorelease];
     ```

  2. Open the PDF documents that contain the pages you want to merge.

```
// File path
NSString *path1 = @"...";
NSURL *url1 = [NSURL fileURLWithPath:path1];
CPDFDocument *document1 = [[[CPDFDocument alloc] initWithURL:url1]
autorelease];

// File path
NSString *path2 = @"...";
NSURL *url2 = [NSURL fileURLWithPath:path2];
CPDFDocument *document2 = [[[CPDFDocument alloc] initWithURL:url2]
autorelease];
```

3. Merge all the pages from the documents you just opened, and import them into the blank PDF document.

```
[document importPages:nil fromDocument:document1 atIndex:document.pageCount];
[document importPages:nil fromDocument:document2 atIndex:document.pageCount];
```

4. Save the document.

```
// Save path
NSString *path = @"...";
NSURL *url = [NSURL fileURLWithPath:path];
[document writeToURL:url];
```

The sample code above allows you to merge all the pages from the two documents. If you're looking to merge or add specific pages from one document to another, you can use `importPages` of `CPDFDocument::importPages:fromDocument:atIndex:` to set specific pages.

- Extract Pages

  `CPDFDocument` can extract ranges of pages from one document and put them into a blank document. If you run this operation, you can effectively extract a PDF as you require. Refer to split pages for more details.

3.5.2 Page Edit

Page manipulation is the ability to perform changes to pages.

- To delete pages from a PDF document, use function `CPDFDocument::removePageAtIndexSet:`.
- To insert a blank page into a PDF document, use function `CPDFDocument::insertPage:atIndex:`.
- To insert an image as an entire page into a PDF document, use function `CPDFDocument::insertPage:withImage:atIndex:`.
- To insert specific page from one document to another, use function `CPDFDocument::importPages:fromDocument:atIndex:`.

- To move a page to a new location, use function `CPDFDocument::movePageAtIndex:withPageAtIndex:`.
- To exchange the location of two document pages, use function `CPDFDocument::exchangePageAtIndex:withPageAtIndex:`.
- To replace original document pages with new pages from a different document, use function `CPDFDocument::removePageAtIndexSet:` and `CPDFDocument::importPages:fromDocument:atIndex:`.
- To rotate a page in a PDF document, refer to the following method in the `CPDFPage` class.

```
// Rotation on a page. Must be 0, 90, 180 or 270 (negative rotations will be
"normalized" to one of 0, 90, 180 or 270).
// Some PDF's have an inherent rotation and so -[rotation] may be non-zero when a
PDF is first opened.
@property (nonatomic,assign) NSInteger rotation;
```

- To crop a page in a PDF document, refer to the following method in the `CPDFPage` class.

```
/**
 * Sets the bounds for the specified box.
 *
 * @discussion If the box does not exist, this method creates it for you.
 * @see CPDFDisplayBox
 */
- (void)setBounds:(CGRect)bounds forBox:(CPDFDisplayBox)box;
```

3.5.3 Document Information

To edit document information, refer to the following method in the `CPDFDocument` class.

```
typedef NSString *CPDFDocumentAttribute NS_STRING_ENUM;

extern CPDFDocumentAttribute const CPDFDocumentTitleAttribute;            // NSString
containing document title.
extern CPDFDocumentAttribute const CPDFDocumentAuthorAttribute;           // NSString
containing document author.
extern CPDFDocumentAttribute const CPDFDocumentSubjectAttribute;          // NSString
containing document title.
extern CPDFDocumentAttribute const CPDFDocumentCreatorAttribute;          // NSString
containing name of app that created document.
extern CPDFDocumentAttribute const CPDFDocumentProducerAttribute;         // NSString
containing name of app that produced PDF data.
extern CPDFDocumentAttribute const CPDFDocumentKeywordsAttribute;         // NSString
containing document keywords.
```

```
extern CPDFDocumentAttribute const CPDFDocumentCreationDateAttribute;       // NSString
representing document creation date.
extern CPDFDocumentAttribute const CPDFDocumentModificationDateAttribute;  // NSString
representing last document modification date.

/**
 * A dictionary of document metadata.
 *
 * @discussion Metadata is optional for PDF documents. The dictionary may be empty, or
only some of the keys may have associated values.
 */
- (NSDictionary<CPDFDocumentAttribute, id> *)documentAttributes;
- (void)setDocumentAttributes:(NSDictionary<CPDFDocumentAttribute, id>
*)documentAttributes;
```

3.5.4 Extract Images

To extract images from a PDF document, use function `CPDFDocument::extractImageFromPages:toPath:`.

Extracting images from a page is time-consuming, and you are advised to perform this operation asynchronously. In addition, you can use `CPDFDocument::cancelExtractImage:` to cancel the operation.

The code below will grab all images from the first page of the given PDF document:

```
NSURL *url = [NSURL fileURLWithPath:@""];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

NSIndexSet *pages = [NSIndexSet indexSetWithIndex:0];
NSString *imagePath = @"";
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    [document extractImageFromPages:pages toPath:imagePath];
});
```

3.6 Security

ComPDFKit PDF SDK protects the content of PDF documents from unauthorized access like copying or printing. It offers developers a way to encrypt and decrypt PDFs, add a password, insert watermark, and more. For controlling over document security in ComPDFKit PDF SDK, security handlers perform user authorization and sets various permissions over PDF documents.

3.6.1 PDF Permission

A PDF file can have two different passwords set, a permissions or owner password and an open or user password.

A user password (also known as an open password) requires a user to type a password to open the PDF. When you try to open a document with a user password, `CPDFView` will show a password prompt to unlock the document. If you want to open a document with a user password programmatically, you can use the `CPDFDocument::unlockWithPassword:` API.

An owner password (also known as a permissions password) requires a password to change permission settings. When an owner password is set, you can configure a set of permissions. For example, you can configure an owner password and the “printing” permission when saving a document to make sure that users who don’t know that owner password can only print the document, but not modify it.

The PDF specification defines the permissions shown below:

- Printing — print the document.
- High-quality printing — print the document in high fidelity.
- Copying — copy content from the document.
- Document changes — modify the document contents except for document attributes.
- Document assembly — insert, delete, and rotate pages.
- Commenting — create or modify document annotations, including form field entries.
- Form field entry — modify form field entries even if you can't edit document annotations.

To access the corresponding permissions, refer to following methods in the `CPDFDocument` class.

```
/**
 * A Boolean value indicating whether the document allows printing.
 */
@property (nonatomic,readonly) BOOL allowsPrinting;
/**
 * A Boolean value indicating whether the document allows printing in high fidelity.
 */
@property (nonatomic,readonly) BOOL allowsHighQualityPrinting;
/**
 * A Boolean value indicating whether the document allows copying of content to the
Pasteboard.
 */
@property (nonatomic,readonly) BOOL allowsCopying;
/**
 * A Boolean value indicating whether you can modify the document contents except for
document attributes.
 */
@property (nonatomic,readonly) BOOL allowsDocumentChanges;
/**
 * A Boolean value indicating whether you can manage a document by inserting, deleting,
and rotating pages.
 */
@property (nonatomic,readonly) BOOL allowsDocumentAssembly;
```

```
/**
 * A Boolean value indicating whether you can create or modify document annotations,
including form field entries.
 */
@property (nonatomic,readonly) BOOL allowsCommenting;
/**
 * A Boolean value indicating whether you can modify form field entries even if you
can't edit document annotations.
 */
@property (nonatomic,readonly) BOOL allowsFormFieldEntry;
```

- Encrypt

  ComPDFKit's `CPDFDocument` API can generate a password-protected document. You can use `CPDFDocument` to create a new password-protected PDF document on disk based on a current document. The user password prevents users from viewing the PDF. If you specify it, you also need to specify an owner password.

  For example, you can configure an owner password and the "printing" permission when saving a document to make sure that users who don't know that owner password can only print the document, but not modify it.

```
NSURL *url = [NSURL fileURLWithPath:@""];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

NSURL *surl = [NSURL fileURLWithPath:@""];
NSDictionary *options = @{CPDFDocumentOwnerPasswordOption : @"The owner password",
                          CPDFDocumentUserPasswordOption : @"The user password",
                          CPDFDocumentAllowsPrintingOption : @(YES),
                          CPDFDocumentAllowsHighQualityPrintingOption : @(NO),
                          CPDFDocumentAllowsCopyingOption : @(NO),
                          CPDFDocumentAllowsDocumentChangesOption : @(NO),
                          CPDFDocumentAllowsDocumentAssemblyOption : @(NO),
                          CPDFDocumentAllowsCommentingOption : @(NO),
                          CPDFDocumentAllowsFormFieldEntryOption : @(NO)};
[document writeToURL:surl withOptions:options];
```

- Decrypt

  ComPDFKit PDF SDK fully supports the reading of secured and encrypted PDF documents.

  To check whether a document requires a password:

```
NSURL *url = [NSURL fileURLWithPath:@""];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

if (document.error &&
    document.error.code == CPDFDocumentPasswordError) {
    // Password required
}
```

To read a PDF document with password protection, use function `CPDFDocument::unlockWithPassword:`. If the password is correct, this method returns `YES`, a `CPDFDocumentDidUnlockNotification` notification is sent. Once unlocked, you cannot use this function to relock the document.

To remove PDF security, call the `CPDFDocument::writeDecryptToURL:` method:

```
NSURL *url = [NSURL fileURLWithPath:@""];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

NSURL *surl = [NSURL fileURLWithPath:@""];
[document writeDecryptToURL:surl];
```

3.6.2 Watermark

Adding a non-removable watermark to documents can discourage viewers from sharing your content or taking screenshots.

- To access the list of watermarks, use function `CPDFDocument::watermarks`.
- To add a watermark, use function `CPDFDocument::addWatermark:`.
- To remove the watermark, use function `CPDFDocument::removeWatermark:`.
- To update the watermark, use function `CPDFDocument::updateWatermark`.

How to generate a PDF with a watermark on all its pages using the `CPDFDocument` API:

```
NSURL *url = [NSURL fileURLWithPath:@""];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

CPDFWatermark *watermark = [[[CPDFWatermark alloc] initWithDocument:document
type:CPDFWatermarkTypeText] autorelease];
watermark.text = @"test";
watermark.scale = 2.0;
watermark.rotation = 45;
watermark.opacity = 0.6;
watermark.verticalPosition = CPDFWatermarkVerticalPositionCenter;
watermark.horizontalPosition = CPDFWatermarkHorizontalPositionCenter;
watermark.tx = 0.0;
watermark.ty = 0.0;
[document addWatermark:watermark];

NSURL *surl = [NSURL fileURLWithPath:@""];
[document writeToURL:surl];
```

3.6.3 Redaction

Redaction is the process of removing images, text, and vector graphics from a PDF page. This not only involves obscuring the content, but also removing the data in the document within the specified area.

Redaction typically involves removing sensitive content within documents for safe distribution to courts, patent and government institutions, the media, customers, vendors, or any other audience with restricted access to the content. Redaction is a two-step process.

- First, redaction annotations have to be created in the areas that should be redacted. This step won't remove any content from the document yet; it just marks regions for redaction.
- Second, to actually remove the content, the redaction annotations need to be applied. In this step, the page content within the region of the redaction annotations is irreversibly removed.

This means that the actual removal of content happens only after redaction annotations are applied to the document. Before applying, the redaction annotations can be edited and removed the same as any other annotations.

Redacting PDFs programmatically:

- Creating Redactions Programmatically

  You can create redactions programmatically via `CPDFRedactAnnotation`. Use the `quadrilateralPoints` or `bounds` property to set the areas that should be covered by the redaction annotation.

  You also have a few customization options for what a redaction should look like, both in its marked state, which is when the redaction has been created but not yet applied, and in its redacted state, which is when the redaction has been applied. It is impossible to change the appearance once a redaction has been applied, since the redaction annotation will be removed from the document in the process of applying the redaction.

  This is how to create a redaction annotation that covers the specified region on the first page of a document:

  ```
  NSURL *url = [NSURL fileURLWithPath:pdfPath];
  CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
  CPDFPage *page = [document pageAtIndex:0];

  CPDFRedactAnnotation *redact = [[[CPDFRedactAnnotation alloc]
  initWithDocument:document] autorelease];
  redact.bounds = CGRectMake(0, 0, 50, 50);
  redact.overlayText = @"REDACTED";
  redact.font = [UIFont systemFontOfSize:12];
  redact.fontColor = [UIColor redColor];
  redact.alignment = NSTextAlignmentLeft;
  redact.interiorColor = [UIColor blackColor];
  redact.borderColor = [UIColor yellowColor];
  [page addAnnotation:redact];
  ```

- Applying Redactions Programmatically

```
[redact applyRedaction];
```

3.7 Conversion

3.7.1 PDF/A

The conversion option analyzes the content of existing PDF files and performs a sequence of modifications in order to produce a PDF/A compliant document.

Features that are not suitable for long-term archiving (such as encryption, obsolete compression schemes, missing fonts, or device-dependent color) are replaced with their PDF/A compliant equivalents. Because the conversion process applies only necessary changes to the source file, the information loss is minimal.

Converts existing PDF files to PDF/A compliant documents, including PDF/A-1a and PDF/A-1b only.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];
[document writePDFAToURL:url withType:CPDFTypePDFA1a];
```

3.8 PDF Editing

PDF editing provides the ability to change content so that its data can be improved or re-purposed.

- Edit text.
- Remove specific content from existing pages.
- Insert or append new content to existing pages.
- Undo or redo any change.
- Set the text alignment.
- Set the text font style.

3.8.1 Initialize PDF Editing

The following code demonstrates how to do PDF editing initialization.

```
NSURL *url = [NSURL fileURLWithPath:pdfPath];
CPDFDocument *document = [[[CPDFDocument alloc] initWithURL:url] autorelease];

CPDFView *pdfView = [[[CPDFView alloc] initWithFrame:self.view.bounds] autorelease];

// Set the document to display
 pdfView.document = document;

// Begin editing text.
[viewer beginEditingLoadType:CEditingLoadTypeText]
```

3.8.2 Customize the Context Menu

You can rewrite the `menuItemsEditingAtPoint:forPage:` of `CPDFView` in the opened PDF preview to set the options of the context menu, which returns to copy, delete, paste, etc. by default. After completing these options, you can copy, paste, cut, or delete text as in Microsoft Word. The following code will show you how to do this.

```
- (NSArray<UIMenuItem *> *)menuItemsEditingAtPoint:(CGPoint)point forPage:(CPDFPage
*)page {
    NSArray * items = [super menuItemsEditingAtPoint:point forPage:page];
    NSMutableArray *menuItems = [NSMutableArray array];
    if (items)
        [menuItems addObjectsFromArray:items];

    return menuItems;
}
```

3.8.3 Set Text Properties

- You can use `setEditingSelectionFontSize:` of `CPDFView` to set the font size of the currently selected code block or text. The following code will show you how to do this.

```
      [self setEditingSelectionFontSize:14.0];
```

- You can use `setEditingSelectionFontColor:` of `CPDFView` to set the font color of the currently selected code block or text. The following code will show you how to do this.

```
      [self setEditingSelectionFontColor:[CPDFKitPlatformColor redColor]];
```

- You can use `setCurrentSelectionAlignment:` of `CPDFView` to set the alignment of the currently selected code block or text. The following code will show you how to do this.

```
      [self setCurrentSelectionAlignment:NSTextAlignmentCenter];
```

3.8.4 Listen to the Changes of Block Editing

ComPDFKit PDF SDK offers developers an API for programmatic change of editing status, which is easy to adjust related UI effects.

```
/**
 * Called when the editing changes.
 */
- (void)PDFViewEditingOperationDidChanged:(CPDFView *)pdfView;

/**
 * Called when the selected block or selected certain area changes.
 */
- (void)PDFViewEditingSelectStateDidChanged:(CPDFView *)pdfView;

/**
 * Called when the selected text block enters the text-editing status.
 */
- (void)PDFEditingViewShouldBeginEditing:(CPDFView *)pdfView textView:(UITextView
*)textView;

/**
 * Called when the selected text block ends the text-editing status.
 */
- (void)PDFEditingViewShouldEndEditing:(CPDFView *)pdfView textView:(UITextView
*)textView;
```

When selecting a text block or a certain area, you can use `editStatus:` of `CPDFView` to get the current text-editing status. The following is the introduction to each status.

- `CEditingSelectStateEmpty` Does not enter the text-editing status.
- `CEditingSelectStateEditTextArea` Selects a text block without entering the text-editing status.
- `CEditingSelectStateEditNoneText` Enters the text-editing status without selecting text.
- `CEditingSelectStateEditSelectText` Enters the text-editing status and selects text.

3.8.5 How to Redo and Undo

You can use `CPDFViewer`'s class to redo and undo. The following code will show you how to do this.

```
if ([pdfView canEditTextUndo]) {
    [pdfView editTextUndo];
}

if ([pdfView canEditTextRedo]) {
    [pdfView editTextRedo];
}
```

3.8.6 How to Set the Alignment of the Selected Text

You can use `CPDFViewer` to set the PDF text alignment. The following code will show you how to do this.

```
/**
 * Get the alignment of the selected text.
 */
NSTextAlignment alignment = [self editingSelectionAlignment];

/**
 * Set the alignment of the selected text.
 */
[self setCurrentSelectionAlignment:sender.selectedSegmentIndex];
```

3.8.7 How to Set the Font Style of Selected Text

ComPDFKit PDF SDK provides developers with programming to edit the text font style, which is easy to adjust the font of related text blocks.

```
/**
 * Get the font name of the selected text.
 */
- (NSString *)editingSelectionFontName;

/**
 * Set the font name of the selected text.
 */
- (void)setEditingSelectionFontName:(NSString *)fontName;

/**
 * Get whether the selected text is italic.
 */
- (BOOL)isItalicCurrentSelection;

/**
 * Set the selected text is italic.
 */
- (void)setCurrentSelectionIsItalic:(BOOL)isItalic;

/**
 * Get whether the selected text is bold.
 */
- (BOOL)isBoldCurrentSelection;

/**
 * Set the selected text is bold.
 */
- (void)setCurrentSelectionIsBold:(BOOL)isBold;
```

4 Support

4.1 Reporting Problems

Thank you for your interest in ComPDFKit PDF SDK, the only easy-to-use but powerful development solution to integrate high quality PDF rendering capabilities to your applications. If you encounter any technical questions or bug issues when using ComPDFKit PDF SDK for iOS, please submit the problem report to the ComPDFKit team. More information as follows would help us to solve your problem:

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

4.2 Contact Information

Home Link:

https://www.compdf.com

Support & General Contact:

Email: support@compdf.com

Thanks,
The ComPDFKit Team