Kofax SignDoc iOS Foundations

Developer's Guide

Version: 1.6.10

Date: 2019-11-18

KOFAX

Legal Notice

Table of Contents

Preface

Related documentation

The full documentation set for Kofax SignDoc is available at the following location:

https://docshield.kofax.com/Portal/Products/en_US/SD/2.2.1-kjbcp1n42d/SD.htm

Training

Kofax offers both classroom and computer-based training that will help you make the most of your Kofax SignDoc solution. Visit the Kofax website at www.kofax.com for complete details about the available training options and schedules.

Getting help with Kofax products

The Kofax Knowledge Base repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base, go to the Kofax website and select **Support** on the home page.

Note The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).

  Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).

  Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.

- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Chapter 1

Introduction

This framework allows capturing of digital signature on any Apple iPhone or iPad. Some devices allow capture using pressure sensitive pens.

Chapter 2

Compatibility

Supported operating systems:

- iOS 9
- iOS 10

Supported devices:

- Apple iPhone
- Apple iPad
- Supported pens:
- Apple Pencil
- Intuos® Creative Stylus
- Intuos® Creative Stylus 2
- Bamboo™ Stylus Fineline
- Bamboo™ Fineline 2
- Bamboo™ Fineline 3
- Bamboo™ Sketch

Supported IDEs:

- Xcode 9

Chapter 3

What's new

- Improved rendering of points for Wacom pens
- Added support for Bamboo™ Sketch
- Fixed signature timestamp values
- Fixes in signature ISO format
- Fixed Xcode 9.3 compatibility

Chapter 4

How it works

When a signature capture is to happen, a popup dialog will be displayed. The user will sign in the signature capture area from the dialog and press a confirmation button. The dialog responds with a callback, and sends the signature back as a supported format for SignDoc SDK. The application will use that signature data in order to sign a PDF document using SignDoc SDK.

Various options can be set for the signature capture dialog: pen size, color and capture strategies. Each signature capture dialog can be configured in a unique way. The configuration is specific to each signature capture dialog.

Chapter 5

API reference

SDDeviceManager

This class manages the discovery of supported signing pens.

Symbols

`+ startDeviceDiscovery`

Starts the device discovery for all types of supported devices. Currently only bluetooth devices are supported, therefore if you intend to use bluetooth devices do call this method some time before starting the capture process because the device discovery process takes time.

Note Even if only the Apple Pencil is used for signing, do call this method because it will confirm whether an Apple Pencil is connected or not. If the Apple Pencil is not found in the list of connected devices before the capture dialog is shown, the user will not be able to sign with the Apple Pencil in that dialog.

`+ deviceIsConnected`

Returns a boolean value that indicates whether any of the supported devices is connected to the framework.

Note It is recommended to call this method when the app enters foreground in order to have an updated status of the connected devices before the signature process is started.

`+ registerForNotifications:`

Register for notifications. See SDDeviceManagerCallback for more details.

SDDeviceManagerCallback protocol reference

This is the protocol for handling device management.

Symbols

`- deviceConnected:`

Returns the type of device that was connected to the framework. The return values are numbers defined in SPiOSFoundationsAPI.h:

- for all supported Wacom pens
  `IOSFOUNDATIONS_DEVICE_TYPE_WACOM_BLUETOOTH_1`
- for Apple Pencil
  `IOSFOUNDATIONS_DEVICE_TYPE_APPLE_PENCIL_1`

Use this callback if it is interesting to know when exactly a pen was connected to the framework.

This callback will also take place after the "deviceIsConnected" method from the "SDDeviceManager" class. Use this method in order to retrieve connectivity status of supported pen types in cases not covered by the callback.

Note The Apple Pencil does not send notification when it is disconnected or reconnected. The connectivity status can be checked by explicitly calling the "deviceIsConnected" method from the "SDDeviceManager" class.

The Apple Pencil does not send notification when it is disconnected or reconnected. The connectivity status can be checked by explicitly calling the "deviceIsConnected" method from the "SDDeviceManager" class.

`- deviceDisconnected:`

Returns the type of device that was disconnected to the framework. The return values are numbers defined in SPiOSFoundationsAPI.h:

- for all supported Wacom pens
  `IOSFOUNDATIONS_DEVICE_TYPE_WACOM_BLUETOOTH_1`
- for Apple Pencil
  `IOSFOUNDATIONS_DEVICE_TYPE_APPLE_PENCIL_1`

SDSignatureCaptureController protocol reference

This is the class that controls the signature capture. The different initialization methods allow customization of the signature capture dialog.

Symbols

@Deprecated

`- initWithParent:withDelegate:`

Initializes the signature capture controller with a parent view controller and a SDSignatureHandler delegate. The signature dialog is build from a resource file and cannot be modified.

`- initWithParent:withDelegate:backgroundImage:dialogXibs:languages:`

The parent parameter is a view controller that acts as a parent for the signature capture view controller. The delegate is described in SDSignatureHandler. The background image is an UIImage that represents the background of the capture area. The image is centered in the capture area. The caller is responsible for providing a correct image. The dialigXibs is an array of Xib file names representing the signature capture dialog. A sample xib named "MyCustomDialog.xib" is provided. The natural approach is to rename and edit this xib file and use it. The first xib that is loaded successfully from the list is used as signature capture dialog. The languages parameter is used for a more detailed search of the dialog xib.

Using this method, minor customization can be done to the capture dialog xib. For example, changing the images of the predefined buttons, moving and resizing the predefined buttons and capture area and any other changes that do not affect functionality of a control.

Note It is impossible to add new controls or change the way the signature capture dialog appears and disappears using this approach.

`- initWithDelegate:backgroundImage:dialogXibs:languages:`

The delegate is described in SDSignatureHandler. The background image is an UIImage that represents the background of the capture area. The image is centered in the capture area. The caller is responsible for providing a correct image. The dialigXibs is an array of Xib file names representing the signature capture dialog. A sample xib named "MyCustomDialog.xib" is provided. The natural approach is to rename and edit this xib file and use it. The first xib that is loaded successfully from the list is used as signature capture dialog. The languages parameter is used for a more detailed search of the dialog xib.

Use this method when customization needs to be done. This method allows retrieval of the UIViewController object.

`- getViewController:`

Returns the view controller of the signature capture dialog. Attaining the UIViewcontroller object provides great flexibility: custom controls can be added, orientation can be controlled.

The displaying must be implemented.

Note Use this method only if the signature capture dialog was initialized with a parent. The capture area should never be resized after the capture dialog is shown, for example during orientation change.

`- setDialogPosition:`

Sets the end display position of the signature capture dialog. The dialog moves from outside the lower part of the screen to the defined position. Allowed values are:

`IOSFOUNDATIONS_DIALOG_POSITION_CENTER`

`IOSFOUNDATIONS_DIALOG_POSITION_TOP`

`IOSFOUNDATIONS_DIALOG_POSITION_BOTTOM`

The default value is:

`IOSFOUNDATIONS_DIALOG_POSITION_CENTER`

Note Use this method only if you initialized the signature capture dialog with a parent. The defined values are found in SPiOSFoundationsAPI.h.

`- setTitle:`

Populates the title label that is available in the default signature capture dialog. If a xib is used, the

title can be changed directly from there.

`- setPenProperties:`

Sets various pen properties. The pen properties are stored into a dictionary. Each property and its values is described below.

`IOSFOUNDATIONS_PEN_PROPERTIES_PEN_SIZE`

Type: NSNumber

Default: 6

Description: Thickness of the strokes

`IOSFOUNDATIONS_PEN_PROPERTIES_COLOR`

Type: UIColor

Default: blueColor

Description: Color of the strokes

`IOSFOUNDATIONS_PEN_PROPERTIES_CAPTURE_STRATEGIES`

Type: NSArray[NSNumber]

Default:

[IOSFOUNDATIONS_CAPTURE_STRATEGY_APPLE_PENCIL, IOSFOUNDATIONS_CAPTURE_STRATEGY_WACOM_BLUETOOTH_WITH_TOUCH_MANAGER, IOSFOUNDATIONS_CAPTURE_STRATEGY_HAND]

Description: A capture strategy is correlated with one or more pens and might have unique ways of capturing data.

The order used to define the capture strategies works like this: each time a capture dialog is opened, the dialog will use the first capture strategy from the list that can be used at that point in time. For example, if the Apple Pencil capture strategy is the first one in the list, but no Apple Pencil is connected, the capture dialog will jump over the Apple Pencil capture strategy and analyze the next one, until it finds one that is usable.

Note The hand capture strategy is always usable, so there is no point to set it as the first one in a list of more capture strategies, because all the others will never be used. This way, the importance of each capture strategy is determined by its order in the list.

If no capture strategy is usable, signing will not be possible with the current capture dialog.

The capture strategies are described below under Capture strategies.

`IOSFOUNDATIONS_SIGNING_AREA_BORDER_STRATEGY`

Type: NSNumber

Default: IOSFOUNDATIONS_SIGNING_AREA_BORDER_STRATEGY_BEST_DATA

Description: A strategy related to the signature capture behavior when the signature leaves the signing area. The strategies are described below under Signing area border strategies.

Capture strategies

`- IOSFOUNDATIONS_CAPTURE_STRATEGY_HAND:`

Use this capture strategy when signing with any device that behaves like a hand. A popup will appear if multiple touches are detected on the singing area and the singing process will restart.

`- IOSFOUNDATIONS_CAPTURE_STRATEGY_APPLE_PENCIL:`

Use this capture strategy to sign with the Apple Pencil.

`- IOSFOUNDATIONS_CAPTURE_STRATEGY_WACOM_BLUETOOTH_WITH_HAND_WARNING:`

A warning dialog will show up if the user touches the signing area with the hand during the

signature capture. Supported pens: Intuos® Creative Stylus, Intuos® Creative Stylus 2, BambooTM Stylus fineline, BambooTM Fineline 2.

`- IOSFOUNDATIONS_CAPTURE_STRATEGY_WACOM_BLUETOOTH_WITH_TOUCH_MANAGER:`

This capture strategy uses the Wacom touch manager class. The touches are fully controlled by the Wacom framework. Supported pens: Intuos® Creative Stylus, Intuos® Creative Stylus 2, BambooTM Stylus fineline, BambooTM Fineline 2.

Note The defined values are found in SPiOSFoundationsAPI.h

Signing area border strategies

`- IOSFOUNDATIONS_SIGNING_AREA_BORDER_STRATEGY_BEST_DATA:`

This is the default signing area border strategy. The goal of this strategy is to provide the best data with as minimum alteration as possible. If the pen goes beyond the border of the signing area, biometric data is collected as long as the pen does not leave the device screen. Note that asking the signature capture dialog for an image only returns what is actually rendered. This means that samples from this scenario are not available in the signature image, but are available in the biometric data.

`- IOSFOUNDATIONS_SIGNING_AREA_BORDER_STRATEGY_ALERT:`

The goal of this strategy is to provide clean signature data. The impact is that whenever the pen leaves the signing area, a popup will show up, alerting the user that the signature is invalid and completely erasing the current signature.

`- IOSFOUNDATIONS_SIGNING_AREA_BORDER_STRATEGY_TRIM:`

This strategy alters the signature data in only one way: samples that go beyond the borders of the signing area are trimmed to the borders of the signing area.

`- signatureAsBlob:`

Returns the signature in an internal representation suitable for SignDoc Web. This is an internal format. Refer to SignDoc Web documentation for details. If signature is not valid, nil is returned.

`- signatureAsISO19794Simple:`

Returns the signature as serialized ISO 19794-Simple object. This is an internal format. Refer to SignDoc SDK documentation for details. If signature is not valid, nil is returned.

Note Usage of this format is shown in the sample application.

`- signatureAsSVG:`

Returns the signature as SVG. This is an internal format. If signature is not valid, nil is returned.

`- signatureAsUIImage:`

Returns the signature as an UIImage. If signature is not valid, nil is returned.

Note Usage of this format is shown in the sample application.

SDSignatureHandler protocol reference

This is the protocol for handling the result of a signature capture process.

Symbols

`- abortSignature:`

Handle cancelled signature process.

`- eraseSignature:`

Handle erase/redo signature event.

`- handleSignature:withFieldId:`

Handle signature data. Use this callback to retrieve the signature data.

Note Example usage is shown in the sample application.

`- pointHistoryX:y:p:t:`

Use this callback to get access to the signature points. Counting these points one by one will result in the total number of points of the signature. Implementing this callback is optional.

Chapter 6

Linkage

This framework makes use of several resource files which, along with the Framework bundle, need to be added to the Xcode project just like other source files. The Framework does not contain Apple Frameworks (it just references them), thus you have to add the following Frameworks to your Xcode project (Target configuration, tab "Build Phases", section "Link Binary With Libraries"):

- Accelerate.framework
- Security.framework
- Kofax SignDoc iOS Foundations Developer's Guide
- QuartzCore.framework
- OpenGLES.framework
- CoreBluetooth.framework
- CoreGraphics.framework
- CoreFoundation.framework
- CoreText.framework
- libz.dylib

In order to link against SignDocSDK, you need to add the following to your Xcode project:

- libSPSignDoc_4.1.a
- an empty "dummy.cpp" file

On the tab "Build Settings", make sure that you include:

```
-ObjC
```

in the "Other Linker Flags" setting,

```
-Compiler Default
```

in the "C++ Language Dialect" setting and

```
-libc++ (LLVM C++ standard library with C++11 support)
```

in the "C++ Standard Library" setting.

Within the SignDocSDK package, the sample is ready to run. Otherwise "SignDoc iOSFoundations*" and a folder named "SignDoc_SDK" (containing the files from the current SignDoc SDK) must be found in the same directory.