Kofax mobiFlow

iOS Developer's Guide

Version: 6.0.0

Date: 2020-10-23

KOFAX

Table of Contents

Preface.......... 5

Getting help with Kofax products.......... 5

Product documentation.......... 6

Online documentation.......... 6

Offline documentation.......... 6

Chapter 1: Project settings.......... 7

Manual installation.......... 7

Other settings.......... 7

Linkage.......... 8

Resources.......... 9

Chapter 2: Using TISOCREngine type (optional).......... 10

TISOCREngineTypeTess.......... 10

Get the config file for your iOS app.......... 11

Chapter 3: Camera capture flow.......... 13

Example with default parameters.......... 13

Example with configured parameters (recommended method).......... 14

Session parameters.......... 15

License parameters.......... 23

IQA parameters.......... 23

Leveler parameters.......... 25

Live OCR parameters.......... 26

Structured OCR parameters.......... 27

Handling messages, errors and results.......... 28

Result delegate.......... 28

For document type Card only.......... 29

Parsing Driver's License (US/Canada only).......... 30

For document type Check only.......... 32

For document type Check only, with CMC7 MICR.......... 32

For document type Card only and OCR type Credit Card.......... 33

For document type Passport only.......... 33

didOutputVideoFeedResultsForValidations.......... 33

captureDidFail.......... 34

finishedMultiPageCaptureSession.......... 38

generalMessages.......... 38

Security recommendations.......... 39
Chapter 4: Set up a custom capture user interface.......... 40
Use the mobiFlow capture screen.......... 40
Change the dynamic capture colors.......... 41
Change labels and messages.......... 41
Change the text indicators.......... 42
Info screen popup.......... 43
Design or change the UI.......... 43
hintDidChange.......... 44
setInstructionLabelText.......... 44
bringButtonsToForground.......... 44
Accessibility.......... 44
Receive mobiFlow notifications.......... 48
Design or change the Guidelines popup UI.......... 48
CustomInfoScreenView.h file.......... 49
CustomInfoScreenView.mm file.......... 49
Split capture front and back (Check only).......... 51
Captions and messages.......... 52
Reporting issues.......... 53
Guidelines for successful capture.......... 54
Contrast.......... 54
Background homogeneity.......... 54
Lighting.......... 54
Shooting and rotation angles.......... 55
Taking the picture.......... 55
Checks only: Digital row (MICR).......... 55

Preface

This guide describes mobiFlow image capture library, and explains how to use this library to integrate mobiFlow into other iOS apps using the Objective C language and XCode IDE.

The mobiFlow SDK is packaged as a framework that is referenced from your project.

Note the following:

- This SDK is relevant only to the applications running on iOS 9.0 or later.
- Image boundaries detection and image contrast verification is done on video frames with medium quality.
- For checks, digital row detection is done on the captured still image or on the video feed. mobiFlow tries to use the best available image quality under the limitations of memory consumption.
- The library crops the image using a special algorithm for boundaries detection, then binarizes (with 1 channel) from a color image to a B&W image and sets it to TIFF with Group 4 Fax Encoding (CCITT T.6).
- For iOS 10 and above integration, see Linkage.
- An integration sample in Objective-C can be found in mobiFlowShowCase application.

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.

Product documentation

By default, the Kofax mobiFlow documentation is available online. However, if necessary, you can download the documentation to use offline.

Online documentation

The product documentation for Kofax mobiFlow 6.0.0 is available at the following location:

https://docshield.kofax.com/Portal/Products/en_US/mobiFlow/6.0.0-tss0pu9zau/mobiFlow.htm

Offline documentation

To access the documentation offline, download the documentation .zip files from the Kofax Fulfillment Site and extract them on a local drive available to your users.

Chapter 1

Project settings

This chapter describes the settings required to create an iOS project.

Manual installation

You can install mobiFlow project manually. To install the mobiFlow project manually, do the following:

1. Add the following frameworks to the project:
   - libz.tbd
   - libc++.tbd
   - AVFoundation.framework
   - CoreVideo.framework
   - CoreMedia.framework
   - CoreMotion.framework
   - AudioToolbox.framework
   - Photos.framework
   - QuartzCore.framework
   - ImageIO.framework
   - Accelerate.framework
   - CoreGraphics.framework
   - OpenGLES.framework
   - UIKit.framework
   - Foundation.framework
   - AssetsLibrary.framework
2. Add the KofaxmobiFlowWidget.framework to the project.
3. Copy opencv2.framework to your project folder in Finder, then add it to the project in Xcode.

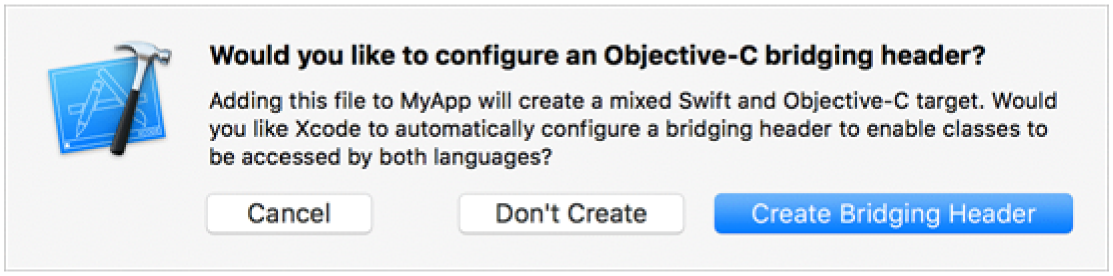
   You can download the opencv2.framework from opencv.org. Use version 3.2.0.
4. In **Build Settings**, make sure you include “$(inherited)” and “$(SRCROOT)” in non-recursive mode under **Framework Search Paths**.

Other settings

Go to **TARGETS** > **General**, and under **Device Orientation**, enable **Landscape Left** and **Landscape Right**.

Linkage

1. Go to the **Build Settings** of your project and scroll down to the **Linking** section. For the property **Other Linker Flags**, add **-ObjC**.
2. For iOS 10 and above integration, add to your .plist file the key Privacy - Camera Usage Description and the value "Used to capture documents". You can edit this value for your own purposes. When using Debug Mode, you will need to also add the key Privacy - Photo Library Usage Description.
3. Go to the **Build Settings** of your project and scroll down to the **Apple LLVM 8.0 – Language – C++** section, then change **c++ Language Dialect** and **c++ Standard Library** to **Compiler Defaults**.
4. (Objective C projects only): Change the file extension of the controller that uses the framework to .mm (not .m), and use #import <KofaxmobiFlowWidgett/TISMobiFlowWidget.h>.
5. (Swift projects only): Skip this step if you already have the Objective-C bridging header.
    a. To import a set of Objective-C files into the same app target as your Swift code, you rely on an Objective-C bridging header to expose those files to Swift. Xcode offers to create this header file when you add an Objective-C file to an existing Swift app.

    Would you like to configure an Objective-C bridging header?

    Adding this file to MyApp will create a mixed Swift and Objective-C target. Would you like Xcode to automatically configure a bridging header to enable classes to be accessed by both languages?

    b. In the Objective-C bridging header, add #import <KofaxmobiFlowWidget/TISMobiFlowWidget.h>.
    c. In your .swift class, declare the delegate TISMobiFlowDelegate and add its functions.
    d. To start a session, add the following code:

```
if let sessionParams = TISSessionParameters(documentType:
 TISDocumentTypeCheck)
{
   if let captureManager = TISCaptureManagerViewController(sessionParameters:
 sessionParams)
   {
   captureManager.captureManagerDelegate = self
   self.present(captureManager, animated: true, completion: nil)
   }
}
```

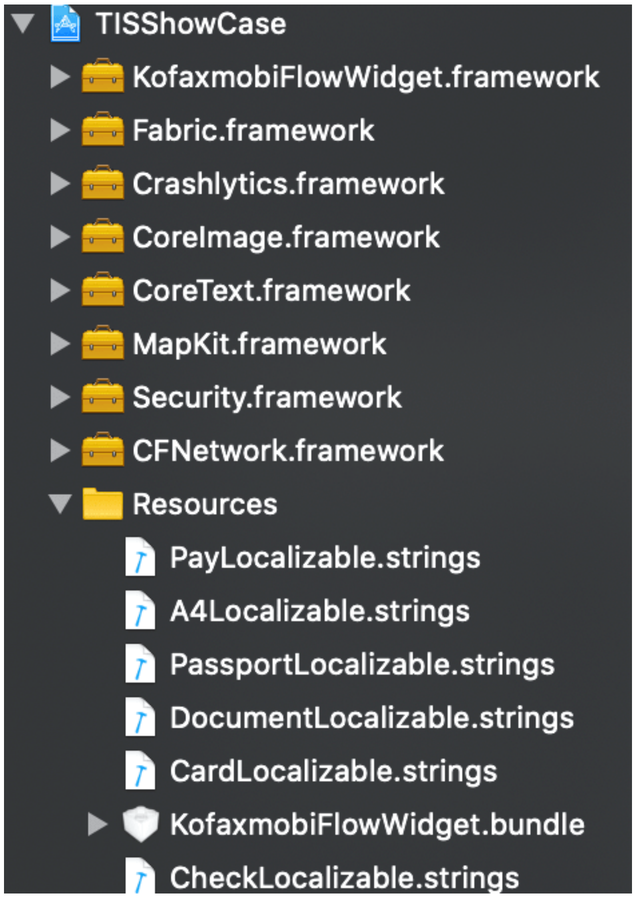
6. You should also change the extension of the files that import this controller to mm, or change the **File Type** to **objective c++ source** in the **Utilities** menu (right pane).

Resources

1. Add the `resources` folder to your project's resources.

   Note KofaxmobiFlowWidget.bundle holds resources vital for the algorithm to function properly. If the bundle is not added correctly, image detection will not work.

2. Add .strings files according to the desired document type, to edit the default app string or change the string to another language.
   Your Xcode project tree should look something like this:



```
▼ TISShowCase
    ▶ KofaxmobiFlowWidget.framework
    ▶ Fabric.framework
    ▶ Crashlytics.framework
    ▶ CoreImage.framework
    ▶ CoreText.framework
    ▶ MapKit.framework
    ▶ Security.framework
    ▶ CFNetwork.framework
    ▼ Resources
          PayLocalizable.strings
          A4Localizable.strings
          PassportLocalizable.strings
          DocumentLocalizable.strings
          CardLocalizable.strings
        ▶ KofaxmobiFlowWidget.bundle
          CheckLocalizable.strings
```

Chapter 2

Using TISOCREngine type (optional)

This chapter describes the types of OCR engines with example codes.

TISOCREngineTypeTess

Add tessdata folder as a reference folder. See the following example code.

Example code for live OCR:

```
TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
initWithDocumentType:
                                              TISDocumentTypeLiveOCR];

sessionParameters.enableLeveler = false;
TISCaptureManagerViewController * captureManagerViewController =
 [[TISCaptureManagerViewController alloc]

 initWithSessionParameters: sessionParameters];

captureManagerViewController.captureManagerDelegate = self;
    [self presentViewController:captureManagerViewController animated:YES
 completion:nil];
```

Example code for structured OCR full page document:

```
TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
 initWithDocumentType:
                                              TISDocumentTypeFullPage];

     TISOCRRegion* rightUpperCornerAmount = [[TISOCRRegion alloc]
 initWithInputROI:CGRectMake(14.5,0.0,6.5,8.0) andRegex:AMOUNT_REGEX];
    TISOCRRegion* rightUpperCornerDate = [[TISOCRRegion alloc]
 initWithInputROI:CGRectMake(14.5,1.0,6.5,2.0) andRegex:DATE_REGEX];
    TISOCRRegion* rightmiddleAmounts = [[TISOCRRegion alloc]
 initWithInputROI:CGRectMake(14.0,9.0,6.0,6.0) andRegex:AMOUNT_REGEX];
    TISOCRRegion* middleRawText = [[TISOCRRegion alloc]
 initWithInputROI:CGRectMake(2.0,13.5,17.0,2.0)];
    CGSize baseSize =CGSizeMake(21.0, 29.5);
    NSMutableArray *arrOCRRegions = [NSMutableArray
 arrayWithObjects:rightUpperCornerAmount,rightUpperCornerDate,
 rightmiddleAmounts,middleRawText,nil];
    TISStructuredOCRParameters* staticOCR = [[TISStructuredOCRParameters alloc]
 initWithOcrRegions:arrOCRRegions andBaseSize:baseSize];
    sessionParameters.structuredOCRSettings = staticOCR;
    sessionParameters.enableLeveler = false;
    sessionParameters.videoFeedProcessing = true;
    sessionParameters.enableStructuredOcrProcessing = true;
    sessionParameters.engineType = TISOCREngineTypeTess;
```

```
TISCaptureManagerViewController * captureManagerViewController =
 [[TISCaptureManagerViewController alloc]

 initWithSessionParameters: sessionParameters];

captureManagerViewController.captureManagerDelegate = self;
    [self presentViewController:captureManagerViewController animated:YES
 completion:nil];
```

To add TISOCREngineTypeML, do the following:

1. Go to https://firebase.google.com.
2. Sign in to your Google account.
3. At the top right of the page, click **Go to console**.
4. Click **Create a project**.
5. Follow the on screen instructions to add your project to console and integrating ML-Kit to your app.

Get the config file for your iOS app

To download the config file for a Firebase iOS app, do the following:

1. Sign into Firebase and open your project.
2. Click ⚙, and select **Project settings**.
3. In the Your apps card, select the bundle ID of the app for which you need a config file.
4. Click ⭳ to download GoogleService-Info.plist, and add it to your app.
5. Include the following ML Kit libraries in your Podfile:

   ```
   pod 'Firebase/Core'
   pod 'GoogleAppMeasurement', '= 5.3.0'
   pod 'Firebase/MLVision'
   pod 'Firebase/MLVisionTextModel'
   ```

   For more information, see Recognize Text in Images with ML Kit on iOS.
6. In build settings under **Other Linker Flags**, add the following:
   - $(inherited)
   - TISOCREngineML.h and TISOCREngineML.mfollowing
   - @import Firebase to AppDelegate.m
   - didFinishLaunchingWithOptions [FIRApp configure];
   - #import "TISOCREngineML.h" to your ViewController.mm

   The following is the example code for live OCR:

   ```
   TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
    initWithDocumentType:
                                                TISDocumentTypeLiveOCR];



       TISOCREngineML *engine = [TISOCREngineML sharedInstance];
       sessionParameters.OCREngineConnector = engine;
   ```

```
    sessionParameters.engineType = TISOCREngineTypeML;
    sessionParameters.enableLeveler = false;
    TISCaptureManagerViewController * captureManagerViewController =
 [[TISCaptureManagerViewController alloc]

 initWithSessionParameters: sessionParameters];
captureManagerViewController.captureManagerDelegate = self;
    [self presentViewController:captureManagerViewController animated:YES
 completion:nil];
```

The following is the example code for structured OCR full page document:

```
TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
 initWithDocumentType:
                                               TISDocumentTypeFullPage];



    TISOCREngineML *engine = [TISOCREngineML sharedInstance];
    sessionParameters.OCREngineConnector = engine;
    sessionParameters.engineType = TISOCREngineTypeML;
    sessionParameters.enableLeveler = false;
    sessionParameters.videoFeedProcessing = true;
    sessionParameters.enableStructuredOcrProcessing = true;
    sessionParameters.uxType = TISFlowUXTypeStatic;
    TISCaptureManagerViewController * captureManagerViewController =
 [[TISCaptureManagerViewController alloc]

 initWithSessionParameters: sessionParameters];
captureManagerViewController.captureManagerDelegate = self;
    [self presentViewController:captureManagerViewController animated:YES
 completion:nil];
```

Chapter 3

Camera capture flow

To launch the camera session, you must create an instance of TISSessionParameters and perform all the changes you want before you initialize TISCaptureManagerViewController.

If you want to use a CustomViewController, you must initialize it before you initialize TISCaptureManagerViewController.

You can initialize TISCaptureManagerViewController using either option:

```
(nullable instancetype) initWithSessionParameters:(nonnull
 TISSessionParameters*)sessionParameters andCustomView:(nullable
 UIViewController*)customViewController;
```

Note With this option user can pass custom view controller.

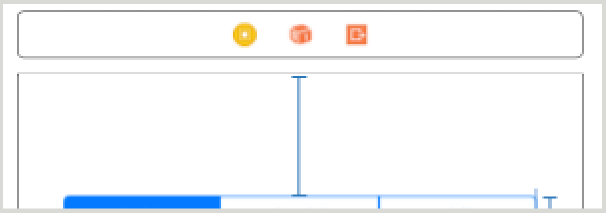
or

```
(nullable instancetype) initWithSessionParameters:(nonnull
 TISSessionParameters*)sessionParameters;
```

Note With this option custom view controller cannot be passed.

The implementation file that contains the reference to TISCaptureManagerViewController should have the extension .mm, not .m.

Note The ViewController that is used to present the camera should not contain a Navigation bar and the top view should be connected to the View and not to the Top Layout Guide. This will make the animation smoother.



Example with default parameters

This example depicts how to run the camera with the default parameters.

```
TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
 initWithDocumentType: TISDocumentTypeCheck];
```

```
TISCaptureManagerViewController * captureManagerViewController =
 [[TISCaptureManagerViewController alloc] initWithSessionParameters:
 sessionParameters];
captureManagerViewController.captureManagerDelegate = self;
[self presentViewController:captureManagerViewController animated:YES completion:nil];
```

Example with configured parameters (recommended method)

This example depicts how to run the camera and configure parameters.

```
TISSessionParameters* sessionParameters = [[TISSessionParameters alloc]
 initWithDocumentType:  TISDocumentTypeCheck ];

//(This is only an example of how to initialize the TISSessionParameters, see the table
 below with all possible values)
sessionParameters.enableIQA = NO;
sessionParameters.showGuidelinesIndicators = YES;
sessionParameters.outputGrayscaleImage = YES;
sessionParameters.outputOriginalImage = YES;
sessionParameters.outputBinarizedImage = YES;
sessionParameters.outputColorImage = YES;
sessionParameters.enableBlurDetection = YES;
sessionParameters.enableCountdownSound = NO;
sessionParameters.enableLeveler = YES;

//Sample Parameters for checks only
sessionParameters.scanFrontOnly = YES;
sessionParameters.ocrType =  OCRType_MICR_E13B ;

//load information view
sessionParameters.showInfoScreen = YES;
sessionParameters.infoScreenInterval = 10.0;

//IQA init will load the default 21 IQA settings
TISCheckIqaParameters* iQAParameters = [[TISCheckIqaParameters alloc] init];

//To load default 51 IQA settings
TISCheckIqaParameters* iQAParameters = [TISCheckIqaParameters IQA51Defaults];

[iQAParameters setCornerFrontSameToAllCorners:0.8f width:0.8f area:0.3f];
[iQAParameters setCornerBackSameToAllCorners:3.0f width:3.0f area:1.0f];
[iQAParameters setEdgeSameToAllSides:0.8f width:0.8f area:0.3f];
[iQAParameters setRotationSkew:7.5f];
[iQAParameters setMaxDarknessBack:0.98f];
[iQAParameters setMaxDarknessFront:0.9f];
[iQAParameters setMinDarknessBack:0.0038f];
[iQAParameters setMinDarknessFront:0.009f];
[iQAParameters setNumberOfSpotsBack:5852];
[iQAParameters setNumberOfSpotsFront:5852];
[iQAParameters setMaxImageFileSizeBack:200.00];
[iQAParameters setMinImageFileSizeBack:0.50];
[iQAParameters setMaxImageFileSizeFront:200.00];
[iQAParameters setMinImageFileSizeFront:0.50];

//This line must be called each time you start a new session
sessionParameters.IQASettings = iQAParameters;

//Leveler
TISLevelerParameters* levelerParameters = [[ TISLevelerParameters alloc] init];

//init will load the default leveler settings
```

```
[levelerParameters setLevelerType:oneUnitLeveler];
[levelerParameters setIsFadeOutEnable:TRUE];
[levelerParameters setIsDraggingEnable:TRUE];
[levelerParameters setLevelerRectSize:150.0f];

//The following initialization can be done for the Two Units Leveler:
[levelerParameters setLevelerType:oneUnitLeveler];
[levelerParameters setLevelerThickness:20.0f];
[levelerParameters setPaddingFromFrame:60.0f];
[levelerParameters setAlignmantToFrame:topRight];

sessionParameters.levelerParameters =  levelerParameters ;

TISCaptureManagerViewController* captureManagerViewController =
 [[TISCaptureManagerViewController alloc] initWithSessionParameters:sessionParameters];

captureManagerViewController.captureManagerDelegate = self;
[self presentViewController:captureManagerViewController animated:YES  completion:nil];
```

A more detailed example is available in the mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Session parameters

Set the parameters for TISSessionParameters according to this table.

Parameter	Description
documentType	Document type set to one of the enums: • TISDocumentTypeCheck • TISDocumentTypeBillPayment • TISDocumentTypeFullPage • TISDocumentTypePassport • TISDocumentTypeCard • TISDocumentTypeCustom • TISDocumentTypeLiveOCR Default: None. You must set this parameter.
debugMode	In debug mode, images are stored on the device and logs are written to the console. Default: NO

Parameter	Description
uxType	Static capture sets predefined boundaries on the screen according to the aspect ratio, while the document must be placed within the shown boundaries. Live capture looks for a quadrilateral of a document in any size, with optional additional settings according to the document type, and validates that the document is in the correct aspect ratio. Setting the aspect ratio to 0.0 both for Minimum and Maximum skips validation in dynamic mode and lets you capture any document. uxType can be set to one of the following enums: • TISFlowUXTypeStatic • TISFlowUXTypeLive Default: TISFlowUXTypeLive
minHeightWidthAspectRatio	Minimum allowed ratio between the height and width of a captured image. Default values: • Checks and bills: 0.35 • Full page: 1.35 • Passport: 0.65 • Card: 0.582
maxHeightWidthAspectRatio	Maximum allowed ratio between the height and width of a captured image. Default values: • Checks and bills: 0.50 • Full page: 1.45 • Passport: 0.8 • Card: 0.7117
enableIQA	Enable or disable the IQA validations. Default: NO
IQASettings	A class of type TISCheckIqaParameters to set all the threshold parameters for the IQA validations. You can leave it to defaults if not in use.
showInfoScreen	Shows the information screen if the user has difficulty capturing the document after a specific set time. Default: YES
InfoScreenInterval	The number of seconds until the information screen appears on the camera overlay. Default: 10.0
showGuidelinesIndicators	When set to NO, only two static indicators are presented. • TISFlowIndicatorAlign: Indicator for alignment (the device should be aligned with the document) • TISFlowIndicatorHold: Indicator for hold (the device should be held over the document) When set to YES, dynamic indicators are presented. Default: YES

Parameter	Description
outputGrayscaleImage	Enables the output of a grayscale JPG. Default: YES
grayscaleImageCompression	A value of the factor by which the JPG cropped grayscale image is compressed. The value ranges from 0.0 for highest compression (lowest quality) to 1.0 (highest quality) Default: 1.0
outputOriginalImage	Enables the output of the captured original image. Default: YES
outputColorImage	Enables the output of the captured cropped color image. Default: YES
colorImageCompression	A value of the factor by which the JPG cropped color image is compressed. Values ranges from 0.0 for highest compression (lowest quality) to 1.0 (highest quality) Default: 1.0
outputBinarizedImage	Enables the output of the captured black and white image. Default: YES
grayScaleSize	Set the width and height of the grayscale output image. The parameter is of typeCG. Default: {0,0}
enableBlurDetection	When set to YES, mobiFlow checks the sharpness of an image and notifies when the image is blurred. **Note** Currently blur detection does not apply on the back side of a document. Default: NO YES for Payment and Full Page, NO for Check. Set to NO for Checks.
videoFeedProcessing	When set to YES, the picture is taken directly from the video feed when the document is aligned properly with the frame. In this case, the device does not switch to still mode and does not present the countdown sequence. When set to NO, the device switches to still mode to take the picture. Must be set to YES for Passport. Default: YES for Check and Passport, NO for other types.
maxVideoFrameToCapture	When video feed processing is enabled, the library tries to process the captured image. In case of failure, this parameter is set to the maximum attempts to capture via video mode before switching back to still mode and countdown. For better performance, set this parameter between 5 and 10. This parameter is relevant only when videoFeedProcessing is set to YES. Default: 7

Parameter	Description
showCountDown	Only applicable to still mode. When set to YES, once the user is in a position to take a picture, the frame turns green and a countdown is shown until the picture is taken automatically. When set to NO, no countdown is shown. The picture is taken when the frame is green and the HOLD STILL message appears on the screen. Default: NO
countDownStartValue	The number from which the countdown starts when the counter for taking a still image is set in this parameter. Default: 2
countDownStopValue	The number at which the countdown stops when the counter for taking a still image is shown. The countDownStopValue must be lower than the countDownStartValue. Default: 0
enableCountdownSound	When set to TRUE, enables a sound along with the image capture countdown. The sound that is played is beepaif from the bundle. Default: NO
dynamicStrings	NSDictionary which enables an alternative dynamic input of strings to be used instead of the checkLocalizable.strings file. Keys to be used in this dictionary are equivalent to the strings name described in Use the mobiFlow capture screen. Default: Nil
showDefaultProcessingView	Shows the processing screen (red spinner). If set to NO, you must implement a custom processing screen using mobiFlow notifications. (See Receive mobiFlow notifications.) Default: YES
surroundingColorForDocumentFrame	The color surrounding the document capture frame. Default: [UIColor colorWithRed:0 green:0 blue:0 alpha:0.8]
enableLeveler	Enables a leveler to be added to the capture frame. The leveler provides visual guidance to the user on how to level the device for successful capture. Default: YES
multiPageCapture	When set to YES, this parameter enables capture of multiple documents. After each capture, a prompt screen is displayed asking user if the user would like to capture another image. If the user selects Finish, the framework calls the finishedMultiPageCaptureSession delegate method. After every captured image, the submitImageResult delegate method is called, but the camera session stays open until the user finishes the multipage session. Default: NO
binarizeBackSameAsFront	When set to YES, the same binarization algorithm that runs on the front side runs on the back side of the check. Default: NO

Parameter	Description
binarizationThreshold	Threshold for the strength of the binarization algorithm. Values can be between 0.0 and 1.0. Set only when capturing a single size document. If the size varies, like Bills, then set to 0.0 for optimization. 1.0: Darkest 0.0: The SDK calculates the optimal threshold according to the image size. Default: 0.0
scanFrontOnly	When set to YES, only the front side is captured. When set to NO and scanBackOnly is set to NO, both front and back are captured.
scanBackOnly	When set to YES, only the back side is captured. If scanFrontOnly is also YES, it fails to initialize the Library. Default: NO
softCapture	Provides the ability to capture the document while the device is held at an angle and not necessarily flat over the document. In this case, the document image is straightened and aligned from the angled position to a flat position. This method may impact the quality of the final image. Default: NO
scanBarcodeLocation	Specify whether to scan the bar code in addition to the document capture session. Specify the side of the document from which capture the bar code. • TISScanBarcodeFront • TISScanBarcodeBack • TISScanBarcodeFrontAndBack • TISScanBarcodeNone Default: TISScanBarcodeNone
showAlertAfterBarcodeRead	Provides an option to show an alert after bar code is detected.

Parameter	Description
barcodeTypes	Relevant only when it is not TISScanBarcodeNone Contains the bar code types that are recognized during the bar code scan session. Once a bar code is detected, if there is a match with one of the bar code types, the bar code is parsed and the SDK continues to capture the document. If one of the bar code types includes TISBarcodeTypeQRCode, TISBarcodeTypeAztecCode, or TISBarcodeTypeDataMatrixCode, a square appears instead of a rectangle for the bar code detection. Supported bar code types in the array: • TISBarcodeTypeUPCECode • TISBarcodeTypeCode39Code • TISBarcodeTypeCode39Mod43Code • TISBarcodeTypeEAN13Code • TISBarcodeTypeEAN8Code • TISBarcodeTypeCode128Code • TISBarcodeTypeCode93Code • TISBarcodeTypePDF417Code • TISBarcodeTypeQRCode • TISBarcodeTypeAztecCode • TISBarcodeTypeInterleaved2of5Code • TISBarcodeTypeITF14Code • TISBarcodeTypeDataMatrixCode Default: All bar code types
ocrType	OCRType enum: • OCRType_MICR_Unknown (For Check only) • OCRType_MICR_E13B (For Check only) • OCRType_MICR_CMC7 (For Check only) • OCRType_MICR_OCRA (For Check only) • OCRType_MRZ (For Passport and Card only) • OCRType_PAN (For Pancard subtype only) • OCRType_CREDIT_CARD (For Card only) • OCRType_OFF Default: OCRType_OFF
minOCRLength (Check document type only)	Minimum number of characters to be recognized. Relevant for check, passport and card. Default values: • Check (MICR) – 15 • Passport (MRZ) – 44 • Card (MRZ) – 30

Parameter	Description
maxOCRLength (Check document type only)	Maximum number of characters to be recognized. Relevant for check, passport and card. Default values: • Check (MICR) – 50 • Passport (MRZ) – 88 • Card (MRZ) – 90
frontImageSize (Check document type only)	Size of the front black and white and grayscale images output. Should be passed as a parameter to the back scan according to the size output of the front scan when the back scan is done separately. See Split capture front and back for more details. The first value in the array is the image width and the second is the image height. Parameter type is Int[].
portraitCapture (Custom document type only)	When set to YES, the camera opens in portrait mode. Default: NO
customROI (Custom document type only)	Sets the region where the frame is displayed in case you need more space for customization. It defined by four parameters, where each parameter is in relation to the screen in terms of x, y, width, height and each value is between 0.0 and 1.0. For example, (0.2, 0.2, 0.5, 0.5) position both x and y at 20% of the full screen from the top left, at a size of 50% of the full screen in height and width. This is only relevant when using uxType static capture. Default: (0.0, 0.0, 1.0, 1.0) – use full screen.
searchForSignature (check document type only)	Verification if there is a signature on the check. Relevant for the front of the check or front and back of the check. From type TISSearchForSignature enum value. When set to TISSignatureNone no signature is searched. Otherwise, directs on which side of the document a signature should be found. Possible values are: • TISSignatureOnFront • TISSignatureOnBack • TISSignatureOnFrontAndBack • TISSignatureNone
liveOCRSettings	Of the type *TISLiveOCR* class, which includes two members that must be initialized. Provides the capability to capture text (with relation to given regex or not) in a specific ROI. There is an option to edit results.
enableStructuredOcrProcessing	Provides the ability to process a whole document or specific regions (if structuredOCRSettings parameter is initialized).
structuredOCRSettings	Relevant when enableOCRProcessing is true. Object of type *TISStructuredOCRParameters* class, which includes two members that must be initialized. Used to provide specific regions to be recognized.

Parameter	Description
OCREngineConnector	This object conforms to TISOCRProcessable protocol. An instance of the OCR engine.
engineType	Whenever live ocr or structured ocr is used you have to set your preferred engine type. TISOCREngineType enum: • TISOCREngineML • TISOCREngineTess Default value : TISOCREngineTess
binarizationType	TISGeneralBinarization: Default value for all document types except Check. TISCheckBinarization: Default value for Check. TISGPUBinarization TISSauvolaBinarization TISOtsuAdaptiveBinarization
license	Of the type TISLicenseParameters class, which includes three members that must be initialized. A valid license must be coded in order for the camera session to start, otherwise a license error message is displayed. See License parameters for more information.
animateTransitionInLivePreview	For TISFlowUXType.LIVE. When set to YES, the green and red rectangles switch with smooth transition animation. Default: YES (BOOL)
softCaptureThreshold	When enabled, the calling app displays the option to control the strictness/ softness of the capture and can allow wider angles and higher capture distance from the frame. Possible values are 0–1. A higher value makes the capture experience less strict. **Note** At the maximum threshold, capture at a wide angle may affect image quality. Default: 0 (float)
tapToFocus	When set to YES, user can tap on the camera overlay to focus explicitly. Default: YES (BOOL)
enableManualCapture	When set to YES, a button is added to the screen, allowing the user to take a still image immediately that will be sent to processing or to the Crop Controller. Default: NO (BOOL)
enableCropController	When set to YES, the image that is taken by manual capture, or automatically by the SDK, is sent to the Crop Controller to confirm the quality and cropping of the image, or to correct the cropping, before it is sent to processing. Default: NO (BOOL)

Parameter	Description
shouldDismissWithAnimation	Dismisses the capture screen with animation. Default: YES (BOOL)
showErrorSignatureOverCMC7 (Check document type only)	When set to YES, ifIf a signature is detected over an CMC7 MICR, sends an error to the calling app. Default: NO (BOOL)

License parameters

Each version of the SDK requires a license. If a license is not configured, mobiFlow displays an error on the device’s screen and does not start the camera session.

The license is individual per implementation and is made up of the licensee name, the license key, and the license itself. The license is either limited by expiration date or is unlimited.

The license is valid per SDK version and can only be used on that version. Upgrading to a newer version requires a new license that matches the version of the SDK used.

You must initialize the following three values (which are provided by mobiFlow).

Parameter	Description
licensee	The name of the licensee that the license is associated with. Usually, this will be the customer name or the project name.
licenseKey	A unique key that is given to each license or customer.
activeLicense	An encrypted string that contains the license information.

Following is the sample code for license.

```
sessionParameters.licenseParams = [[TISLicenseParameters alloc]
 initWithLicensee:@"ABCD" licenseKey:@"a70e52b0-e499-3562-afb1-17f04038356b"
 activeLicense:@"TqeRDhExXuGCLNdIcvb4OR9+QJYiTnWQ3ooFtcWx39OkkNeUYf4Ph0U
+P5x6DaRIdA84HwlWUzF5YMLA5k=="];
```

If the license information is validated successfully, the camera session starts.

If the license validation fails, an error is displayed on the screen to the user and the camera session closes.

IQA parameters

IQA is used to define validation for image quality.

Set the parameters for iQAParameters according to the this table.

Parameter	Description
RotationSkew	Maximum skewing angle allowed.

Parameter	Description
minDarknessFront	Minimum ratio of black pixels to total pixels for the front side.
maxDarknessFront	Maximum ratio of black pixels to total pixels for the front side.
minDarknessBack	Minimum ratio of black pixels to total pixels for back side.
maxDarknessBack	Maximum ratio of black pixels to total pixels for back side.
numberOfSpotsFront	Maximum number of spots that are considered as spots allowed per square inch on average for the front side. Black areas count as spots if the size of the area is greater than 3 pixels and less than 20 pixels, and the black area is surrounded by white pixels.
numberOfSpotsBack	Maximum number of spots that are considered as spots allowed per square inch on average for the back side. Black areas count as spots if the size of the area is greater than 3 pixels and less than 20 pixels, and the black area is surrounded by white pixels.
CornerDataArrayFront	Thresholds for height, width and area (in inches) for every corner of the check on the front side. Use the function setCornerFrontSameToAllCorners to set the same height, width and area for all corners, or use SetCornerFrontAll to set a different threshold for each corner.
CornerDataArrayBack	Thresholds for height, width and area (in inches) for every corner of the check on the back side. Use the function setCornerBackSameToAllCorners to set the same height, width and area for all corners, or use SetCornerBackAll to set a different threshold for each corner.
EdgeDataArray	Thresholds for height, width and area (in inches) for every side of the check (top/bottom/left/right). Use the function setEdgeSameToAllSides to set the same height, width and area to all corners, or use SetEdgeAll to set different threshold for each corner.
MinImageFileSizeFront	The minimum file size for the TIFF image for the front side.
MaxImageFileSizeFront	The maximum file size for the TIFF image for the front side.
MinImageFileSizeBack	The minimum file size for the TIFF image for the back side.
MaxImageFileSizeBack	The maximum file size for the TIFF image for the back side.
horizontalStreakSumOfBlackPixels	The minimum number of black pixels required to determine if the line is black.
horizontalStreakLineWidth	The minimum width of the black line to detect.
horizontalStreakNumLines	The minimum number of black lines for the horizontal streaks alert .
carbonStripSumOfBlackPixels	The minimum number of black pixels required to determine if the line is black.
carbonStripLineWidth	The minimum width of the black line to detect.
carbonStripNumLines	The minimum number of black lines for the horizontal streaks alert.
piggyBackMaxWidth	Maximum width threshold between two checks that overlap each other.
piggyBackMaxHeight	Maximum height threshold between two checks that overlap each other.

Parameter	Description
piggyBackMaxAR	Maximum aspect ratio between the two checks.
piggyBackMinAR	Minimum aspect ratio between the two checks.

Leveler parameters

Set the parameters for LevelerParameters according to this table.

Parameter	Description
levelerType	Defines the leveler type. Possible values: • oneUnitLeveler • twoUnitsLeveler • scaleLeveler Default value: scaleLeveler
isFadeoutEnabled	Relevant for all leveler types. Defines whether the leveler should fade out when the device is leveled. Default value: YES
isDraggingEnabled	Relevant for all leveler types. Enables the user to drag the leveler on the screen. Default value: YES
levelerRectSize	Relevant for the oneUnitLeveler type only. The size of the leveler. The leveler rectangle size range is between 80.0 and the height of the capturing frame. Default value: 150.0
levelerRectCenter	Relevant for the oneUnitLeveler type only. Indicates the location of the leveler on the screen. Default value: the center of the capturing frame.
alignmentToFrame	Relevant for twoUnitLeveler and scaleLeveler only. The alignment of the two leveler units to the capturing frame: • topLeft • bottomLeft • topRight • bottomRight Default value: topRight
levelerThickness	Relevant for twoUnitLeveler and scaleLeveler only. The thickness of the leveler unit's frames. The leveler thickness ranges between 10.0 and 30.0. Default value: 10.0

Parameter	Description
paddingFromFrame	Relevant for twoUnitLeveler and scaleLeveler only. The distance of the leveler unit's rectangles from the capturing frame. The leveler padding range is between 25.0 and a maximum padding value that is dynamically calculated by the following formula:(Capturing frame width/height – leveler minimum size)/2 Default value: 25.0 **Note** The padding is from both sides of the capturing frame and therefore its value is multiplied by 2.
levelerDisplay	Relevant for the scaleLeveler type only. Defines where the leveler is to be presented on the screen, using the enum TISScaleLevelerDisplay: • TISScaleLevelerShowBothScales • TISScaleLevelerShowHorizontalScale • TISScaleLevelerShowVerticalScale • TISScaleLevelerShowNone Default value: TISScaleLevelerShowHorizontalScale
scaleUnitGap	Relevant for the scaleLeveler type only. The distance between the leveler’s units. The number of units are dynamically calculated accordingly. Default value: 60.0
userColorsInScale	Relevant for the scaleLeveler type only. Customizes the scale leveler and set its colors. It can be set to multiple colors or a single color. The array should be initializes in this form: (A color, B color ,…,B color, A color). There should be a minimum of one object in the array. Default value: white

Live OCR parameters

Set the parameters for LiveOCRParameters according to this table.

Initialize the following two values.

Parameter	Description
arrLiveOcrParameters	An array of with objects of type *TISLiveOcrParameters*. This array sets the number of fields for live recognition. Each object contains: • title: The name of field to recognize. • scanType: • ScanType_Date: A regex for recognize date is provided. • ScanType_Amount: A regex for recognize date is provided. • ScanType_RawText: It recognizes texts. • ScanType_Custom: For custom regex use. • width: The width of the mask rectangle shown on screen to capture specific ROI (value between 0.0 to 1.0). • height: The height of the mask rectangle shown on screen to capture specific ROI (value between 0.0 to 1.0). • customROI: The mask rectangle calculated by given width and height in relation to screen dimensions. • regex: Optional regular expression for specific ROI. • detectedSentence: Initialized when recognition is done.
ocrConfig	Each object contains autoDetectorOcr which determines if after recognition it will pass automatically to next ROI for ocr.

Structured OCR parameters

This object is relevant only if enableStructuredOCRProcessing is true.

This capability enables the developer to insert specific regions to be recognized.

Set the parameters for structuredOCRParameters according to this table.

Parameter	Description
ocrRegions	An array of with objects of type *TISOCRRegion*. Each object contains: • inputROI: The specific ROI to be process. Needs to be measured by developer relative to the property ocrBaseSize. • resultROI: Available after process - used by Kofax where there is recognition - the 'inputROI' converted to output image coordinates system. • regex: Optional, output only text that meets this regular expression. • regionConfidence: Available when OCR Engine type is TISOCREngineTess. • ocrProcessResults: Contains all the OCR results in the form of array. When this array is not initialized OCR will apply on the whole document.
ocrBaseSize	The width and height that all regions are derived from (the size of the document to process).

Handling messages, errors and results

To get camera session results, set TISMobiFlowDelegate and implement the methods didFinishWithResults and cancel.

Result delegate

Following is the signature of TISMobiFlowDelegate.

```
(void) captureManager:(TISCaptureManagerViewController *)captureManagerViewController
 didFinishWithResults:(TISProcessingResults *)imageResults
```

This method is called once the image is captured successfully.

The TISProcessingResults class contains the following properties with the results.

Parameter	Description
originalFront	The JPEG representation of the front original image.
originalBack	The JPEG representation of the back original image.
tiffFront	The TIFF representation of the front image.
tiffBack	The TIFF representation of the back image.
grayscaleFront	The grayscale JPEG image of the front side of the image.
grayscaleBack	The grayscale JPEG image of the back side of the image.

Parameter	Description
colorFront	The color JPEG image of the front side of the image.
colorBack	The color JPEG image of the back side of the image.
barcodeResult	A dictionary that contains two or four objects in the following format: • TISBarcodeType Front, bar code parsed string for Front. • TISBarcodeType Back, bar code parsed string for Back. If scanBarcodeLocation is set to TISScanBarcodeNone an empty dictionary is returned.
captureManagerViewController	A reference to the TISCaptureManagerViewController.
arrOcrResults	An array which contains TISOCRResult objects. The TISOCRresult object contains the following: • text: The text that the OCR engine has recognized for this block. • boundingBox: The bounding box rectangle where this recognized block appears in the cropped image. • confidence: Available for TISOCREngineTess only. The confidence in the accuracy of this recognition result. This number will be between 0.0 and 100.0. • inputROI: If TISOCRRegion was set, then this property refers to the same inputROI as in TISOCRRegion. • resultROI: Result region of interest, this is available after image processing. • inputRegex: Optional, the regex that was used to recognize text.

For document type Card only

The TISCardProcessingResults class inherits from TISProcessingResults and contains the following properties with the results.

Parameter	Description
IDCardResultsByField	A dictionary with the MRZ recognition’s results. This dictionary has the following fields: • kTISCard_Type • kTISCard_IssuingCountry • kTISCard_DocumentNumber • kTISCard_DateOfBirth • kTISCard_Sex • kTISCard_ExpirationDate • kTISCard_Nationality • kTISCard_Surname • kTISCard_FirstName • kTISCard_MiddleName

Parsing Driver's License (US/Canada only)

When configuring scanBarcodeLocation for front / back / front+back, and barcodeTypes contains TISBarcodeTypePDF417Code, the SDK can parse the bar code results of a US/Canadian driver’s license.

The dictionary includes the following keys:

- First Name
- Middle Name
- Last Name
- Name Suffix
- Address
- City
- State
- Postal Code
- ID Number
- Class
- Height
- Weight
- Eye Color
- Hair Color
- Expiration Date
- Date Of Birth
- Sex
- Issue Date
- Restriction Code
- Endorsement Code
- Limited Duration Document Indicator
- Document Number

- Country ID
- Inventor Control Number
- Card Revision Date
- Temp Visitor
- Address
- Address Additional info
- Duplicates
- Organ Donor
- Audit Information
- Ethnicity
- Compliance Type
- First Name Truncation
- Middle Name Truncation
- Last Name Truncation
- Federal Commercial Vehicle Code
- Customer Specific Control Number
- WA Specific Endorsements
- Transaction Types
- Under 18 Until
- Under 21 Until
- Revision Date
- Social Security Number

To retrieve the dictionary, use the following methods:

```
NSDictionary *parsedDLResult = [TISDLBarcodeParser
parseDLBarcodeWithString:[imageResults.barcodeResult
objectForKey:BARCODE_DATA_FRONT]];
```

Note The above method is used when user has raw content of bar code string.

or

```
NSDictionary *parsedDLResultB = [imageResults.barcodeResult
objectForKey:BARCODE_PARSED_DATA_FRONT];
```

Note The above method is used when user has parsed content of bar code string.

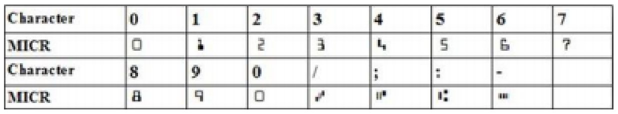
The keys can be as follows:

Parameter	Description
BARCODE_TYPE_FRONT	The type of bar code capture for the front side.
BARCODE_TYPE_BACK	The type of bar code capture for the back side.
BARCODE_DATA_FRONT	The raw data of the bar code for the front side.
BARCODE_DATA_BACK	The raw data of the bar code for the back side.

Parameter	Description
BARCODE_PARSED_DATA_FRONT	The parsed driver's license data for the front side.
BARCODE_PARSED_DATA_BACK	The parsed driver's license data for the back side.

For document type Check only

The TISCheckProcessingResults class inherits from TISProcessingResults and contains the following properties with the results.

Parameter	Description
result	The MICR result formatted in mobiFlow format (special characters represented by a dash).
rawResult	The result of every character in the MICR is represented by a number, separated by commas. 0,1,2,3,4,5,6,7,8,9,10,12,11,13 The numbers represent the MICR in the order given in the following table: 
resultsScores	The score for each of the recognized characters separated by commas, respective to the rawResult.

Character	0	1	2	3	4	5	6	7
MICR	0	1	2	3	4	5	6	7
Character	8	9	0	/	;	:	-	
MICR	8	9	0	⑉	⑈	⑆	⑇	

```
- (void)
captureManager:(TISCaptureManagerViewController*)captureManagerViewController
didFinishWithResults:(TISProcessingResults*)imageResults
{
   if ([imageResults isKindOfClass:[TISCheckProcessingResults class]])
   {
      NSString *caption;
      NSString *micrResult = [(TISCheckProcessingResults*)imageResults
 getFormattedMICRString:captureManagerViewController.sessionParameters.ocrType];
      if (micrResult.length)
      {
         self.strMicr = [NSString stringWithFormat:@"Check MICR is %@", micrResult];
caption = [NSString stringWithFormat:@"Front Original Colored Jpeg image,
%@", self.strMicr];
      }
   }
}
```

For document type Check only, with CMC7 MICR

The TISCMC7CheckProcessingResults class inherits from TISCheckProcessingResults and contains the following properties with the results.

Parameter	Description
signatureOverCMC7MicrDetected	Indicates if a signature was detected on the CMC7 MICR.

For document type Card only and OCR type Credit Card

The TISCreditCardProcessingResults class inherits from TISProcessingResults and contains the following properties with the results:

Parameter	Description
numbers	The credit card numbers. This value is always extracted from the credit card.
expiryMonth	The expiry month of the credit card. This value is not found always, if it is not found, the value will be 0.
expiryYear	The expiry year of the credit card. This value is not found always, if it is not found, the value will be 0.

For document type Passport only

The TISPassportProcessingResults class inherits from TISProcessingResults and contains the following properties with the results.

Parameter	Description
passportResultsByField	Dictionary with the passport’s results. The dictionary contains the following keys: • kTISPassport_Type; • kTISPassport_IssuingCountry; • kTISPassport_Surname; • kTISPassport_FirstName; • kTISPassport_PassportNumber; • kTISPassport_Nationality; • kTISPassport_DateOfBirth; • kTISPassport_Sex; • kTISPassport_ExpirationDate; • kTISPassport_PersonalNumber;

didOutputVideoFeedResultsForValidations

The signature of didOutputVideoFeedResultsForValidations is as follows:

```
-(BOOL) captureManager:(TISCaptureManagerViewController*)captureManagerViewController
didOutputVideoFeedResultsForValidations:(TISProcessingResults*)imageResults;
```

This optional method for the delegate TISMobiFlowDelegate is called when OCR results were detected on the video feed for each successful frame that passed mobiFlow internal validations, for Check or Passport document type and Pan Card subtype.

The main use of this method is to allow the hosting app to run additional validations on the raw OCR results. Then the hosting app can return YES if the results are valid, or NO to continue the video feed processing of other frames and get another result.

Parameter	Description
TISCaptureManagerViewController	A reference to the TISCaptureManagerViewController.
TISProcessingResults	Contains the OCR results, as detailed in the TISProcessingResult section.
captureManagerViewController.validationType	This property is of the type TISFlowValidationType. Using this parameter, the developer can distinguish between the types of validations that needs to be applied. Since two OCRs can be chosen in the same session, such as MICR and structured OCR, this method will be called twice in this scenario; in every call developer must verify which validation is needed at the moment. Possible values for TISFlowValidationType are the following: • TISFlowMICRValidation • TISFlowMRZValidation • TISFlowOCREngineValidation • TISFlowPanCardValidation

Return value.

Value	Description
BOOL	Return YES if the results are valid, or NO to continue the video feed processing and get another result.

A more detailed example can be found in the Kofax mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Important Whenever you set micr/mrz/pan ocr in the session parameters AND enableStructuredOcrProcessing is on, you MUST use the validationType enum.

captureDidFail

The signature of captureDidFail is as follows:

```
- (BOOL) captureManager:(TISCaptureManagerViewController*)captureManagerViewController
captureDidFail:(TISCaptureErrorCode)TISErrorCode;
```

This method is optional to implement. It informs of an error and allows the delegate to handle the error and how the library should handle the error.

Parameter	Description
TISFlowErrorCode	TISCaptureErrorCode enum: • TISFlowErrorGeneralFail • TISFlowErrorOCRReading • TISFlowErrorImageContrast • TISFlowErrorNoValidBoundingBox • TISFlowErrorIQACornerData • TISFlowErrorIQAEdgeData • TISFlowErrorIQASkew • TISFlowErrorIQADarkness • TISFlowErrorIQANumSpots • TISFlowErrorBlurDetected • TISFlowErrorMICRLength • TISFlowWarningMICRInterupted (Only for CMC7) • TISFlowWarningMICRDetectedOnCheckBack • TISFlowErrorLicenseInvalid • TISFlowErrorLicenseExpired • TISFlowErrorHorizontalStreaks • TISFlowErrorCarbonStrip • TISFlowErrorPiggybackFound
captureManagerViewController.captureResults	This property is of the type TISProcessingResults. If an error occurs or the SDK fails for some reason, all available output is returned.

Return value:

Value	Description
BOOL	Return YES for error handling in the delegate. This will close the library and return control to the calling app. Return NO for error handling to take place in the mobiFlow framework.

The following sample codes demonstrate the captureDidFail implementation options:

First option: SDK handles errors

```
- (BOOL) captureManager:(TISCaptureManagerViewController *)
 captureManagerViewController
captureDidFail:(TISFlowErrorCode)TISErrorCode
{
return NO;
}
```

Second option: Close the camera when receiving an error

```
- (BOOL) captureManager:(TISCaptureManagerViewController *)
captureManagerViewController captureDidFail:(TISFlowErrorCode)TISErrorCode
{
   [captureManagerViewController.cameraOverlayViewController closeCamera];
   return YES;
}
```

Third option: Host app handles the error and the session continues to another retry

The error can be specific (this example is for iOS 8):

```
- (BOOL) captureManager:(TISCaptureManagerViewController *)
captureManagerViewController captureDidFail:(TISFlowErrorCode)TISErrorCode
{
if (TISErrorCode == TISFlowErrorNoValidBoundingBox){//TISFlowErrorMICRReadingCheck
dispatch_sync(dispatch_get_main_queue(), ^{

if(SYSTEM_VERSION_LESS_THAN(@"8.0"))
{
// UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Custom Error Message"
message:@"TISFlowErrorMICRReadingCheck" delegate:self cancelButtonTitle:@"OK"
otherButtonTitles:nil, nil];
[UIAlertView showWithTitle:@"Custom Error Message"
message:@"TISFlowErrorNoValidBoundingBox" cancelButtonTitle:@"OK" otherButtonTitles:nil
tapBlock:^(UIAlertView *alertView, NSInteger buttonIndex) {
[captureManagerViewController.cameraOverlayViewController restartVideoSession];
} ];
}
else
{
UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"Custom Error Message"

message:@"TISFlowErrorMICRReadingCheck"

preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *okAction = [UIAlertAction
actionWithTitle:@"OK"
style:UIAlertActionStyleDefault
handler:^(UIAlertAction *action)
{
[captureManagerViewController.cameraOverlayViewController
restartVideoSession];
}];
[alertController addAction:okAction];
[captureManagerViewController.cameraOverlayViewController
presentViewController:alertController animated:YES completion:nil];
}
    });
   return YES;
}
else if (TISErrorCode==TISFlowWarningMICRInterupted)
{
dispatch_after(dispatch_time(DISPATCH_TIME_NOW, 1.5 * NSEC_PER_SEC),
dispatch_get_main_queue(), ^(void){
if ([captureManagerViewController.captureResults
isKindOfClass:[TISCheckProcessingResults class]]) {
if(SYSTEM_VERSION_LESS_THAN(@"8.0"))
{
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"MICR Interrupted"
message:[NSString stringWithFormat:@"The digital line is in bad quality or interrpted
 by
signature.\nplease check
MICR:%@",[(TISCheckProcessingResults*)captureManagerViewController.captureResults
result]] delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
[alert show];
     }
     else
     {
```

```
UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"MICR Interrupted"
message:[NSString
stringWithFormat:@"The digital line is in bad quality or interrpted by signature.
\nplease check
MICR:%@",[(TISCheckProcessingResults*)captureManagerViewController.captureResults
result]]

preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *okAction = [UIAlertAction
actionWithTitle:@"OK"
style:UIAlertActionStyleDefault
handler:^(UIAlertAction *action)
{
[captureManagerViewController.cameraOverlayViewController restartVideoSession];

}];
[alertController addAction:okAction];
[captureManagerViewController.cameraOverlayViewController
presentViewController:alertController animated:YES completion:nil];
}
     }
});
return YES;
}
return NO;}
}
```

A more detailed example can be found in the Kofax mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

The order in which the validations run is different when using stills mode and video mode, and so are the messages that are used. The following table shows the order of the validations and their application per document type and capture mode.

Validation description	**Validation error code (enum)**	**Error message name**	**Display message on video feed processing**	**Message on stills**
Image Contrast	TISFlowErrorImageContrast	TISFlowErrorImageContrast	NO*	YES
Blur Detection**	TISFlowErrorBlurDetected	TISErrorBlurFail	NO*	YES
Look For Document Rectangle	TISFlowErrorNoValid BoundingBox	TISFlowErrorNoValid BoundingBox	NO*	YES
User is capturing the front side instead the of back side of the check***	TISFlowWarningMICR DetectedOnCheckBack	TISFlowWarningMICR DetectedOnCheckBack	YES	YES
OCR Validation	TISFlowErrorOCRReading Check	TISFlowErrorReading Message	YES	YES
MICR Length Validation***	TISFlowErrorMICRLength	TISFlowDigitalRowNotIn Scope	YES	YES
MICR Line Interruption By Signature.CMC7 Only***	TISFlowWarningMicr Interrupted	TISFlowWarningMicr Interrupted	YES	YES

Validation description	Validation error code (enum)	Error message name	Display message on video feed processing	Message on stills
IQA Folded Corner***	TISFlowErrorIQACornerData	TISFlowErrorIQACornerData	YES	YES
IQA Folded Edge***	TISFlowErrorIQAEdgeData	TISFlowErrorIQAEdgeData	YES	YES
IQA Skew***	TISFlowErrorIQASkew	TISFlowErrorIQASkew	YES	YES
IQA Darkness***	TISFlowErrorIQADarkness	TISFlowErrorIQADarkness	YES	YES
IQA Number of Spots***	TISFlowErrorIQANumSpots	TISFlowErrorIQANumSpots	YES	YES
IQA Horizontal Streaks***	TISFlowErrorHorizontal Streaks	TISFlowErrorHorizontal Streaks	YES	YES
IQA Carbon Strip***	TISFlowErrorCarbonStrip	TISFlowErrorCarbonStrip	YES	YES
IQA Piggy Back***	TISFlowErrorPiggyback Found	TISFlowErrorPiggyback	YES	YES

* When no message is thrown, mobiFlow proceeds to process the next frame.

** Enabled on documents without OCR.

*** Checks only.

Note IQA validations are performed only for Checks and black and white images.

finishedMultiPageCaptureSession

```
- (void) finishedMultiPageCaptureSession:(TISCaptureManagerViewController*)
captureManagerViewController;
```

This delegate method is called when user selects the Finish button when asked to capture another image when isMultiPage is set to YES.

Parameter	Description
TISFlowGerneralMessagesCode	TISFlowGerneralMessagesCode enum: TISFlowMessageBarcodeRead

A more detailed example can be found in the Kofax mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

generalMessages

```
- (void) captureManager:(TISCaptureManagerViewController*)captureManagerViewController
 generalMessages:(TISFlowGerneralMessagesCode)TISGeneralMessageCode;
```

This delegate method is called when the SDK informs about actions.

A more detailed example can be found in the Kofax mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Security recommendations

The mobile calling application has the responsibility to protect the data returned by the SDK in the downstream flow until the mobile application is closed. The mobile calling application implementing the mobiFlow SDK should adhere to security best practices in order to protect any sensitive data and customer information.

Some of the considerations while implementing and configuring the SDK are the following:

- The mobile calling application is responsible for ensuring that any sensitive data received from the SDK process follows existing processes for safeguarding the data. It is assumed that whatever processes are used for manually entered data would be applied to data extracted from the SDK process.
- On closing the SDK, images and/or data are erased from memory. It is the responsibility of the mobile calling application to ensure that the SDK is closed and objects are released upon completion of the SDK process.
- When debugMode is set to TRUE, images captured by the SDK are stored on the device (as well as the logs). It is strongly recommended that IsDebug always be set to FALSE in the release mode of the application build (Production code), as the images and application data should not be physically stored outside the context of the mobile application. The images and data should only exist in the temporary memory of the mobile application and should not be accessible outside the application context.
- For on-device OCR of Checks (for account funding use cases), it is not recommended to return the check image to the user. Only the extracted data should be returned. To do this, set the output settings to **FALSE**:
  - outputGrayscaleImage = FALSE
  - outputOriginalImage = FALSE
  - outputColorImage = FALSE
  - outputBinarizedImage = FALSE
- The Android SDK documentation provides additional code samples (saveImagesToDevice()) to save the images on the device after retrieving them from the SDK. Similar code may also be implemented for iOS as well. It is the responsibility of the application team to ensure any images/data available from the SDK are not stored on the device, especially in the release mode of the application (Production code).This code should only be used for testing and troubleshooting issues in the development cycle.

Chapter 4

Set up a custom capture user interface

This chaper explains how to customize the capture user interface (UI).

There are two options when implementing the library:

- Keeping the same capture screen that mobiFlow provides, and changing the logo, icons and captions.
- Designing your own UI or hiding some controls in the mobiFlow screen by inheriting and overriding the current UI.

Be aware that you are unable to change the frame control because it includes all the functionality for automatically capturing the image.

Use the mobiFlow capture screen

To use this method, you need to change some files in the `resources\KofaxmobiFlowWidget.bundle` folder.

To change icons in the capture screen do the following:

1. Find the KofaxmobiFlowWidget.bundle file in Finder and remove the .bundle suffix.
2. Open the KofaxmobiFlowWidget directory, and replace the desired file.
3. Add the .bundle suffix to the KofaxmobiFlowWidget directory.
4. Compile and run.

You must include the rest of the files that you did not change in the new bundle.

File name	Description
logoWatermark.png	The logo of the company.
btnTorch.png	The flash icon when not selected.
btnTorchSelected.png	The flash icon when selected.
beep.aiff	The sound to be played along with the image capture countdown. The sound will only be played if EnableCountdownSound is set to YES.

Note Each icon should also have an X2 version for the Retina display version.

You can also change the icons of the indicators and the frame (Static capture only).

File name	Description
boundaryBottom.png	The bottom-right boundary of the frame when document is not found.
boundaryTop.png	The top-left boundary of the frame when document is not found.
boundaryBottomV.png	The bottom-right boundary of the frame when document is found.
boundaryTopV.png	The top-left boundary of the frame when document is found.
boundaryBottomLeft.png	The bottom-left boundary of the frame when document is not found.
boundaryTopRight.png	The top-right boundary of the frame when document is not found.
boundaryBottomLeftV.png	The bottom-left boundary of the frame when document is found.
boundaryTopRightV.png	The top-right boundary of the frame when document is found.

Each icon should also have an X2 version for the retina display version.

Important Do not change any other images or files in this folder.

Change the dynamic capture colors

To change the color of the dynamic overlay rectangle when TISFlowUXTypeLive is used, you must set the TISOverlayDynamicRectangleColors TISDynamicRectangleColors parameter with the desired colors.

The array app should fill this array in this specific order with four UIColor objects:

```
[validRectStrokeColor, validRectFillColor, invalidRectStrokeColor,
 invalidRectFillColor]
```

The following example includes the default colors to show how this can be done (default colors are green frame and fill for valid, and red frame and clear fill for invalid).

```
TISCaptureManagerViewController *captureManagerViewController =
 [[TISCaptureManagerViewController alloc] initWithSessionParameters:sessionParams];
…
captureManagerViewController.cameraOverlayViewController.TISOverlayDynamicRectangleColors
 = [NSArray arrayWithObjects:[UIColor colorWithRed:0.480 green:0.754 blue:0.234
 alpha:1.000],[UIColor colorWithRed:0.675 green:0.853 blue:0.505 alpha:0.500],[UIColor
 colorWithRed:0.914 green:0.058 blue:0.214 alpha:0.500], [UIColor clearColor],  nil];
//defaults colors
```

Change labels and messages

To change the caption of the message and the label on the top, you must change the messages in the Localizable.strings per language and document.

The relevant messages to change are as follows.

String Name	Description
TISFlowPleaseCaptureImage	The label’s caption at the top of the capture screen
TISFlowPleaseCaptureImageBack	The label’s caption at the top of the capture screen (for back)
TISSuccessfulReadingTitle	For combined front and back capture, the title of the message that is displayed after successful capture of the front
TISSuccssfulReadingMessage	For combined front and back capture, the contents of the message that is displayed after successful capture of the front
TISFlowPleaseCaptureBarcode	Instruction for the user to capture the bar code with Static capture, when bar code capture is enabled

Change the text indicators

In the Localization files per document and language, change the relevant string:

String name	Description
TISFlowIndicatorAlign	Indicator to hold the device flat over the check
TISFlowIndicatorDown	Indicator to move the device towards the bottom of the check
TISFlowIndicatorLeft	Indicator to move the device left
TISFlowIndicatorRight	Indicator to move the device right
TISFlowIndicatorTop	Indicator to move the device towards the top of the check
TISFlowIndicatorRotateLeft	Indicator to rotate the device left (check is at an angle)
TISFlowIndicatorRotateRight	Indicator to rotate the device right (check is at an angle)
TISFlowIndicatorZoomIn	Indicator to move closer to the check (check is too far from the frame)
TISFlowIindicatorZoomOut	Indicator to move away from the check (check is exceeding the frame)
TISFlowIndicatorLight	Indicator to turn on the flash (there is not enough light)
TISFlowIndicatorHold	Indicator to hold the camera when the check is found, before the picture is taken
TISFlowScanBarcode	Indicator to move towards the bar code
TISFlowPleaseCaptureCreditCard TISFlowIndicatorScanCreditCard	Indicator to align with the credit card boundaries
TISFlowInvalidRotation	Indicator that phone and document do not have the same orientation

Info screen popup

The info screen popup is displayed when the user has difficulties capturing the document after a certain time (customizable).

The popup will animate from the top screen to the center, in landscape mode.

Change the text in the localization file.

String name	Description
TISFlowInfoScreenText	The Text in the instructions screen.
TISFlowInfoScreenTitle	The Title of the instructions screen.
TISFlowInfoScreenButtonCaption	The Caption of the close button.
TISFlowInfoScreenCheckBoxCaption	The Text of the checkbox caption.

Design or change the UI

To implement your own UI, changing the locations of the mobiFlow control or hiding mobiFlow controls, you must create a new class in your implementation that inherits TISCaptureViewController.

1. Create a new class.
2. In the new class, import TISCaptureViewController.h and rename your .m file to .mm.
3. Implement the method (void) viewDidLoad.
4. In this method, call [super viewDidLoad].
5. Include your implementation.

You can use the instructions in the Use the mobiFlow capture screen section to change icons and messages as you need, and the changes will apply in this method as well.

The mobiFlow library has a few UI controls where the controls' properties are exposed and can be set from the viewDidLoad method.

The following UI controls are available.

UI control name	Description
counterLabel	Count down label
counterImage	Count down image
btnTorch	Flash button
btnCancel	Cancel button
instructionsLabel	Instructions label
hintLabel	Indicator label
watermark	Logo image

You can also write your own code to add new controls to the screen, for example, if you want to add other labels, pictures or buttons.

If you choose to hide the original Cancel button (which is not recommended), you must implement a call to the mobiFlow Close action from the main class; this is essential for the proper functioning of the library. When creating your Close button, you will allocate a method to handle the click action on the button. From this method, you will need to create a call to [self cancelAction], and then implement the rest of your implementation for the action.

hintDidChange

Implement hintDidChange when the hint that shows on the screen changes. This method is fired and you can set different properties to the UI elements including, but not limited to, text, accessibility settings, color, fonts, and so on.

setInstructionLabelText

Override this method if you want to customize the string that will be inserted to the instruction label. When you have finished customizing the string, or when no customization is needed, add the string to the instruction label using self.instructionsLabel.text = text or by calling [super setInstructionLabelText:text].

If you change the string to NSMutableAttributedString, you must change the label text using [self.instructionsLabel setAttributedText: attributedInstruction], (do not call super).

bringButtonsToForground

This method brings the UI to the foreground every time the session is restarted.

If you use a custom view, you must override this method to bring your UI to the foreground as well. Call super if you also use the default UI.

Accessibility

mobiFlow exposes all the elements in the view and allows changing any properties of the elements. This means that the accessibility properties of these elements can be changed by the hosting app in runtime when using custom view.

A sample code for creating such class can be:

CustomView.h file

```
#import <UIKit/UIKit.h>
#import <KofaxmobiFlowWidget/TISMobiFlowWidget.h>
@interface CustomView : TISCaptureViewController
@end
```

CustomView.mm file

```
#import "CustomView.h"
@implementation CustomView
{
CGRect frameRect;
}
- (void)viewDidLoad
```

```
{
[super viewDidLoad];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(receiveTISNotification:)
name:TIS_PROCESS_NOTIFICATION
object:nil];
frameRect =[[UIScreen mainScreen] bounds];

[self hideParentButtons];

//Change instructionLabel position and UI
self.instructionsLabel.textColor = [UIColor redColor];
CGRect frame = self.instructionsLabel.frame;
frame.origin.y += 30;
self.instructionsLabel.frame = frame;

//Add bottom banner image
UIImage *bottom_image = [UIImage imageNamed:@"banner_bottom.png"];
_banner_bottom = [[UIImageView alloc] initWithImage:bottom_image];
[_banner_bottom setFrame:CGRectMake(0, frameRect.size.height-bottom_image.size.height,
 frameRect.size.width, bottom_image.size.height)];
[_banner_bottom setUserInteractionEnabled:YES];
[self.view addSubview:_banner_bottom];

//Add cancel button
_btnCancelOverlay = [UIButton buttonWithType:UIButtonTypeCustom];
UIImage *cancel_image = [UIImage imageNamed:@"cancel_btn.png"];
[_btnCancelOverlay setImage:cancel_image forState:UIControlStateNormal];
[_btnCancelOverlay setFrame:CGRectMake(_banner_bottom.frame.size.width-
cancel_image.size.width-10,
(_banner_bottom.frame.size.height-cancel_image.size.height)/2.0,
cancel_image.size.width,
cancel_image.size.height)];

[_btnCancelOverlay addTarget:self action:@selector(customCancelAction:)
 forControlEvents:UIControlEventTouchUpInside];
[_banner_bottom addSubview:_btnCancelOverlay];

if (self.sessionParameters.enableManualCapture)
{
//Add auto capture button
_autoCaptureButton = [[UIButton alloc] initWithFrame:CGRectMake(10.0, 5.0, 80.0,
 _banner_bottom.frame.size.height - 10.0)];
_autoCaptureButton.layer.borderWidth = 2.0;
_autoCaptureButton.layer.borderColor = [UIColor blackColor].CGColor;
[_autoCaptureButton setTitle:@"Auto On" forState:UIControlStateNormal];
[_autoCaptureButton setTitleColor:[UIColor blackColor] forState:UIControlStateNormal];
[_autoCaptureButton addTarget:self action:@selector(customToggleAutoCaptureAction)
 forControlEvents:UIControlEventTouchUpInside];
[_banner_bottom addSubview:_autoCaptureButton];

//Move manual capture button position
CGRect frameBtnNew = CGRectMake(frameRect.size.width -
 self.btnManualCapture.frame.size.width - 5.0, (frameRect.size.height -
 self.btnManualCapture.frame.size.height)/2.0, self.btnManualCapture.frame.size.width,
 self.btnManualCapture.frame.size.height);
self.btnManualCapture.frame = frameBtnNew;
}

//Add top banner image
UIImage *top_image = [UIImage imageNamed:@"banner_top.png"];
_banner_top = [[UIImageView alloc] initWithImage:top_image];
[_banner_top setFrame:CGRectMake(0, 0, frameRect.size.width, top_image.size.height)];
[self.view addSubview:_banner_top];
```

```
//Add description label
descriptionLabel = [[UILabel alloc] initWithFrame:CGRectMake(0,
 frameRect.size.height/2+20, frameRect.size.width, 50)];
[descriptionLabel setBackgroundColor:[UIColor clearColor]];
[descriptionLabel setTextColor:[UIColor whiteColor]];
[descriptionLabel setFont:[UIFont systemFontOfSize:16]];
[descriptionLabel setTextAlignment:NSTextAlignmentCenter];
[descriptionLabel setNumberOfLines:2];
[descriptionLabel setText:@"Center your bill stub here and we will capture\n the
 information"];
[self.view addSubview:descriptionLabel];

//Adding Activity Indicator for Processing stage
_indicator = [[UIActivityIndicatorView
 alloc]initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhiteLarge];
_indicator.frame = CGRectMake((frameRect.size.width - _indicator.frame.size.width)/2.0,
(frameRect.size.height - _indicator.frame.size.height)/2.0 ,
 _indicator.frame.size.width, _indicator.frame.size.height);

[self.view addSubview:_indicator];

self.TISOverlayDynamicRectangleColors = [NSArray arrayWithObjects:
[UIColor blueColor],
[UIColor clearColor],
[UIColor yellowColor],
[UIColor clearColor],  nil];

//you can add or remove the grid view as well
//uncomment if you want to remove the grid
//[self removeTisGrid];

}
-(void) hideParentButtons
{
if (self.sessionParameters.enableManualCapture) {
self.btnAutoCapture.hidden = YES;
}

self.btnCancel.hidden = YES;
self.watermark.hidden = YES;
self.btnTorch.hidden = YES;
self.counterLabel.hidden = YES;
}
-(void) setInstructionLabelText:(NSString*)text
{
if ([text isEqualToString:NSLocalizedStringFromTable(@"TISFlowPleaseCaptureImage",
 @"CheckLocalizable", "")])
{
NSRange boldRange = [text rangeOfString:@"front"];
if (boldRange.location == NSNotFound)
{
[super setInstructionLabelText:text];
}
else
{
NSMutableAttributedString *attributedInstruction = [[NSMutableAttributedString alloc]
 initWithString:text];
[attributedInstruction addAttribute: NSFontAttributeName value:[UIFont
 boldSystemFontOfSize:18] range:boldRange];
[self.instructionsLabel setAttributedText: attributedInstruction];
}
}
```

```
else if([text
 isEqualToString:NSLocalizedStringFromTable(@"TISFlowPleaseCaptureImageBack",
 @"CheckLocalizable", "")])
{
NSRange boldRange = [text rangeOfString:@"back"];
if (boldRange.location == NSNotFound)
{
[super setInstructionLabelText:text];
}
else
{
NSMutableAttributedString *attributedInstruction = [[NSMutableAttributedString alloc]
 initWithString:text];
[attributedInstruction addAttribute: NSFontAttributeName value:[UIFont
 boldSystemFontOfSize:18] range:boldRange];
[self.instructionsLabel setAttributedText: attributedInstruction];
}
}
else
{
[super setInstructionLabelText:text];
}
}
- (void)hintDidChange:(HintTypeIndicator)hint
{
//    [super hintDidChange:hint];

self.hintLabel.textColor = [UIColor whiteColor];
self.hintLabel.backgroundColor = [UIColor redColor];
self.hintLabel.alpha = 1.0;
self.hintLabel.font = [UIFont boldSystemFontOfSize:16];
[self.hintLabel setCenter:CGPointMake(frameRect.size.width/2,
 frameRect.size.height/2)];
UIAccessibilityPostNotification(UIAccessibilityAnnouncementNotification,
 self.hintLabel.text);
}
-(void)bringButtonsToForeground
{
[self.view bringSubviewToFront:_banner_top];
[self.view bringSubviewToFront:_banner_bottom];
[self.view bringSubviewToFront:_autoCaptureButton];

[super bringButtonsToForeground];
}
#pragma mark - override methods
//Uncomment to implement
//-(void) initDisplay
//{
//
//}
//-(void)initMustHaveDisplayElements
//{
//    [super initMustHaveDisplayElements];
//}
-(void)customCancelAction:(id)sender
{
[self cancelAction:(id)sender];
}
-(void)customToggleAutoCaptureAction
{
[_autoCaptureButton setTitle: self.isBtnAutoCaptureToggleOn ? @"Auto Off" : @"Auto On"
 forState:UIControlStateNormal];

[self toggleAutoCapture];
```

```
}
@end
```

In order for the mobiFlow library to use your custom view, add your custom view to your project (for example, CustomView.h).

Example:

```
…
TISSessionParameters* sessionParameters =[[TISSessionParameters alloc]
 initWithDocumentType:TISDocumentTypeCheck];

CustomView* myView = [[CustomView alloc] init];

TISCaptureManagerViewController* captureManagerViewController =
 [[TISCaptureManagerViewController alloc] initWithSessionParameters:sessionParams
 andCustomView:myView];

captureManagerViewController.captureManagerDelegate = self;
[self presentViewController:captureManagerViewController animated:YES  completion:nil]
```

Receive mobiFlow notifications

When the countdown sequence starts or when image processing starts or ends, mobiFlow sends the following notifications to any registered observers:

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(receiveTISNotification:)
name:TIS_PROCESS_NOTIFICATION
object:nil];
```

Each mobiFlow notification includes information about the event that triggered the notification.

You can access this information from the userInfo NSDictionary.

- When implementing the Cancel button, you should disable its action when getting TISNotificationStatusCountDownStarted and enable it again on captureDidFail, or when processing is finished if you are planning to capture another document in the same session. Refer to the customView class in the sample app for more information.
- A more detailed example can be found in the Kofax mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Design or change the Guidelines popup UI

To implement your own Guidelines Popup UI, changing the locations of the mobiFlow control or hiding mobiFlow controls, you must create a new class in your implementation that inherits TISInfoScreenView.

1. Create a new class.
2. In the new class, import <KofaxmobiFlowWidget/TISInfoScreenView.h>, and rename your .m file to .mm.

3. Implement one of the following methods:
   - -(instancetype) initWithFrame:(CGRect)frame
   - -(instancetype)initWithFrame:(CGRect)frame andDocType:(TISDocumentType)docType
   - -(instancetype)initWithIsPortraitCapture:(BOOL)portraitCapture andDocType:(TISDocumentType)docType
4. In this method, call to super according to the method you implemented, such as [super initWithFrame:frame].
5. Include your implementation.

The mobiFlow library has a few UI controls where the properties of the controls are exposed and can be set from the initWithFrame method.

The following UI controls are available.

Controls name	Description
infoTxtTitle	Title label
textField	Text to be shown
btnClose	Close button
checkBox	Check box button
checkBoxLabel	Check box label

You can also write your own code to add new controls to the screen, for example, if you want to add other labels, pictures or buttons. If you choose to hide the original Cancel button (which is not recommended), you must implement a call to the mobiFlow Cancel action from the main class; this is essential for the proper functioning of the library. When creating your Cancel/Back button, you will allocate a method to handle the click action on the button. From this method, you will need to create a call to [self cancelAction], and then implement the rest of your implementation for the action.

If you choose to hide the original checkBox button (which is not recommended), you must implement a call to the mobiFlow dontShowAgain action from the main class; this is essential for the proper functioning of the library. When creating your checkBox button, you will allocate a method to handle the click action on the button. From this method, you will need to create a call to [self showAgainAction:(bool)toShow], and then implement the rest of your implementation for the action.

Following are the sample codes for creating such class.

CustomInfoScreenView.h file

```
#import <UIKit/UIKit.h>
#import <KofaxmobiFlowWidget/TISInfoScreenView.h>
@interface CustomInfoScreenView : TISInfoScreenView
@end
```

CustomInfoScreenView.mm file

```
#import "CustomInfoScreenView.h"

@implementation CustomInfoScreenView

-(id) initWithFrame:(CGRect)frame
```

```
{
if((self = [super initWithFrame:frame]))
{
self.infoTxtTitle.hidden = YES;
self.textField.hidden = YES;
self.checkbox.hidden = YES;
self.checkboxLabel.hidden = YES;
self.btnClose.hidden = YES;

//adding custom close button
UIButton *btnOverlay = [UIButton buttonWithType:UIButtonTypeCustom];
[btnOverlay setBackgroundColor:[UIColor blueColor]];
[btnOverlay setTitle:@"Close Button" forState:UIControlStateNormal];
[btnOverlay.titleLabel setFont:[UIFont boldSystemFontOfSize: 15.0]];
[btnOverlay setFrame:CGRectMake(10, 225, 100, 30)];
[btnOverlay setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];
[btnOverlay addTarget:self action:@selector(customAction:)
forControlEvents:UIControlEventTouchUpInside];
[self addSubview:btnOverlay];

//adding custom dont show again button
UIButton *dontShowAgain = [UIButton buttonWithType:UIButtonTypeCustom];
[dontShowAgain setBackgroundColor:[UIColor blueColor]];
[dontShowAgain setTitle:@"Dont Show Again Button" forState:UIControlStateNormal];
[dontShowAgain.titleLabel setFont:[UIFont boldSystemFontOfSize: 15.0]];
[dontShowAgain setFrame:CGRectMake(135, 225, 180, 30)];
[dontShowAgain setTitleColor:[UIColor whiteColor] forState:UIControlStateNormal];
[dontShowAgain addTarget:self action:@selector(dontShowAgain)
forControlEvents:UIControlEventTouchUpInside];
[self addSubview:dontShowAgain];

//adding custom instruction label
UILabel *uiLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 20, 440, 150)];
[uiLabel setBackgroundColor:[UIColor clearColor]];
[uiLabel setFont:[UIFont boldSystemFontOfSize: 18.0]];
uiLabel.numberOfLines = 4;
[uiLabel setTextColor:[UIColor whiteColor]];
uiLabel.text = @"TIPS:\n1. Lay bill on dark surface. \n2. Fit entire bill in guides.
\n3.
Hold phone flat.";
[self addSubview:uiLabel]
}
return self;
}
-(void)customAction:(id)sender{
[self closeAction];
}
-(void)dontShowAgain{
[self showAgainAction:NO];
[self closeAction];
}
@end
```

A more detailed example can be found in the mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

In order for the mobiFlow library to use your custom view do the following:

1. Add your custom view to your project (for example, CustomInfoScreenView.h)
2. Add the two lines shown in the following code to the code for initializing TISCaptureManagerViewController (see Camera capture flow). Make sure that the rectangle in the initialization is the final size and location of the popup.

```
…
CustomInfoScreenView *infoScreen = [[CustomInfoScreenView alloc]
initWithFrame:CGRectMake(10, 10, 460, 300)];

checkCaptureManagerViewController.cameraOverlayViewController.infoScreenView =
infoScreen;
```

A more detailed example can be found in the mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Split capture front and back (Check only)

To separately capture the front side and back side of the check in separate sessions:

1. Launch the camera session with scanFrontOnly set to YES in the TISSessionParameters.
2. Launch the camera session again with scanBackOnly set to YES in the TISSessionParameters, and the MICR type set to OCRType_OFF and frontImageSize, as retrieved from the didFinishWithResults delegate of the front capture.

Front capture

```
- (IBAction)scanFront:(id)sender {

TISSessionParameters* checkSessionParameters = [[TISSessionParameters alloc]
 initWithDocumentType:TISDocumentTypeCheck];

//Manual Settings
checkSessionParameters.isDebug=NO;
checkSessionParameters.scanFrontOnly=YES;
checkSessionParameters.ocrType =  OCRType_MICR_E13B ;

TISCaptureManagerViewController
*checkCaptureManagerViewController=[[TISCaptureManagerViewController alloc]
 initWithSessionParameters: checkSessionParameters];
checkCaptureManagerViewController.captureManagerDelegate=self;

[self presentViewController:checkCaptureManagerViewController animated:YES
 completion:nil];
}
```

Back capture

```
- (IBAction)scanBack:(id)sender {
TISSessionParameters* checkSessionParameters =[[TISSessionParameters alloc]
 initWithDocumentType:TISDocumentTypeCheck];
checkSessionParameters.scanBackOnly=YES;
checkSessionParameters.ocrType=OCRType_OFF;

//the frontImageSize can be retrieved from the result of frontTiff.size
checkSessionParameters.frontImageSize = savedFrontImageSize;
TISCaptureManagerViewController
 *checkCaptureManagerViewController=[[TISCaptureManagerViewController alloc]
 initWithSessionParameters: checkSessionParameters];
checkCaptureManagerViewController.captureManagerDelegate=self;
[self presentViewController:checkCaptureManagerViewController animated:
YES completion:nil];
}
```

A more detailed example can be found in the mobiFlow ShowCase app sample, which is included in the SDK Bundle package.

Captions and messages

The relevant Localization files are available in the resources in different languages. You can change the captions and messages used in the Library during the process.

The relevant messages are described in the following table:

Message Name	Description
TISFlowPleaseCaptureImage	Caption displayed when capturing the image/check on the preview screen.
TISFlowPleaseCaptureImageBack	Caption displayed when capturing the back side of the check.
TISFlowCancel	The Cancel button shows in the error messages.
TISFlowOK	The OK button shows in the error messages.
TISFlowPleaseCaptureBarcode	Instruction for the user to capture the bar code in Static capture, when bar code capture is enabled.
TISFlowDigitalRowNotInScope (Checks only)	Message when the digital row is not within the length in the settings.
TISFlowErrorReading	Title for all error messages.
TISFlowErrorReadingMessage	Message when reading the OCR in stills mode failed, recapture of the front is needed.
TISFlowErrorImageContrast	Message when there are contrast issues in detecting colors on the image in stills mode.
TISFlowErrorReadingGeneral	General message about failure to validate the image; issued if a more specific message does not apply.
TISFlowErrorNoValidBoundingBox	Message when the rectangle of the image was not detected by the Library. Message when the bounding box of the image was not detected by the Library.
TISFlowErrorIQACornerData	Message when one of the corners of the check is missing and over the accepted threshold.
TISFlowErrorIQAEdgeData	Message when one of the edges of the check is missing and over the accepted threshold.
TISFlowErrorIQASkew	Message when the check is skewed over the accepted threshold.
TISFlowErrorIQADarkness	Message when the image is too dark over the accepted threshold.
TISFlowErrorIQANumSpots	Message when the image has too much noise and the number of spots per square inch exceeds the accepted threshold.
TISFlowErrorFileTooSmall	Message when the file generated by the Library is too small, below the minimum accepted threshold.
TISFlowErrorMinImageDimensions	Message when the image is not within the dimensions or aspect ratio that is expected.

Message Name	Description
TISFlowErrorUnknown	Message about IQA validation failure, issued if a more specific message does not apply.
TISErrorBlurFail	Message when the image is detected as blurry.
TISFlowWarningMICRDetectedOnCheckBack (Checks only)	Message when the MICR was detected while the user tried to capture the back of the check (meaning they were capturing the front of the check instead of the back).
TISFlowWarningMicrInterrupted (Checks only)	Message when the recognition of the MICR detects that there is something interrupting the MICR recognition, such as stains or the signature. Works on Checks with CMC7 MICR line only.
TISFlowMultiCaptureTitle	Title of the message for multi-capture.
TISMultiCaptureShouldContinueCapture	Message to check whether the user wants to capture another document.
TISFlowFinish	The caption on the button to finish multi-capture.
TISFlowCapture	The caption on the button to continue and capture another document.
TISFlowCancel	The caption of the Cancel button on alerts.

Reporting issues

To report issues to Kofax, you must reproduce the issue on the mobiFlow Showcase app, setting the debug mode ON.

When the debug mode is ON, images and logs are saved on the device for debugging purposes. These images and logs can be sent to the Kofax Support Team to enable them to investigate any issues or bugs that you may encounter. In debug mode, every image that is captured is saved, even if you receive an error message after the capture.

To access these images and logs, you need a program on your computer that can explore the file system of your device when it is connected to the computer via USB. An example of such an app is iFunbox, which can be downloaded from the Internet for free.

To access these images and logs, you need an app on the device that can explore the file system. An example of such an app is ES File Explorer, which can be downloaded from the Google Play Store.

The images are saved on the device under the relevant user application (if using the Showcase app, this will be TISShowcase) in a folder named DEBUG under the Documents folder.

```
<user application>/TISShowcase/Documents/DEBUG
```

The images are saved on the device under the root folder in a folder named .mobiflow (`<root>/.mobiflow`) and the log file is saved in a folder named .debugmobiflow (`<root>/.debugmobiflow/log.txt`). These two folders are hidden, so you will need to go on the app you are using to browse the file system and enable viewing of hidden folders.

As there is only one log file, it grows with every capture. Therefore it is important to delete it before logging something that you want to report. Make sure the log contains only the data from the relevant capture you had issues with.

For every capture, all four images for the front and four for the back will be saved, depending on which images you decided to output (see Handle messages, errors and results).

When reporting an issue, please send the following to Kofax:

- The log file containing only the issue you are reporting.
- All relevant images regarding the issue.
- A detailed description of the issue and step-by-step instructions on how to reproduce.
- Information about the device or devices and the operating system of the device relevant to the issue.
- Information about the Showcase or SDK version relevant to the issue.
- The configuration of all the parameters in the Showcase or SDK where the issue occurs.

If the issue is related to difficulties in capture, and you were not able to capture the document, you can take a picture of the document with your native camera app on the device and send it to the Support Team instead. It would also be very helpful if you can scan the document on a proper scanner and send a copy that the Support Team can print and test themselves.

Guidelines for successful capture

To ensure successful and optimal capture from the mobiFlow library, you should include the following guidelines in your application’s instructions, which should be followed before the user starts the capture process. These guidelines are not mandatory, and a document can still be captured even if the guidelines are not followed, but following them will ensure optimal capture and the best result.

Contrast

The document should be positioned on a background with a different color. Strong visual contrast near the document’s boundaries is particularly advised. For documents with multiple colors around the boundaries, the document background should be a different color from any color on the document’s boundaries.

Background homogeneity

The background should be clean and homogenous. In particular, strong lines on the background that do not belong to the document should be avoided. It is best to have the surface around the document clear of any objects about 6” (15 cm) from each side of the document.

Lighting

Strong direct sunlight or artificial lighting on the document is not recommended. In particular, having strong light on one part of the document and shade over another part should be avoided at all times. Such a situation can result in an unusable black and white image of the part that is not in the shade.

Shooting and rotation angles

The phone's camera should be positioned as flat as possible relative to the document's surface. Moreover, the in-plane rotation of the camera should be similar to that of the document, that is, the picture should be taken in landscape. The document should be positioned in the center of the screen, within the displayed frame and as close as possible to the frame sides.

Taking the picture

When the HOLD STILL message appears, device should be held still over the document until the countdown is over and the still picture is taken. Moving or shaking during this process may result in a blurry image and leads to a failure or an unclear black and white image.

Checks only: Digital row (MICR)

Make sure that the digital row is clean and the signature is not stretching over it. Ensure that all the digits and special characters are readable.