Top Image Systems

Digital Business
Starts Here.

mobiFLOW

iOS Library SDK Guide

Version 5

contact@topimagesystems.com

www.topimagesystems.com

11 January, 2018

Top Image Systems

Digital Business
Starts Here.

Contents

About this guide 5

General notes 5

Project settings 6

Installation 6

Manual installation 6

CocoaPods installation 7

Other settings 7

Linkage 7

Resources 8

Camera capture flow 9

Example with default parameters 10

Example with configured parameters (recommended method) 10

Session parameters 13

License parameters 23

IQA parameters 24

Leveler parameters 27

Handling messages, errors and results 29

Result delegate 29

For document type Card only 30

Parsing Driver's License (US/Canada only) 31

For document type Check only 32

For document type Check only, with CMC7 MICR 33

For document type Card only and OCR type Credit Card 34

For document type Passport only 34

didOutputVideoFeedResultsForValidations 34

captureDidFail 35

finishedMultiPageCaptureSession 41

generalMessages 42

Security recommendations 42

Set up a custom capture user interface 43

Use the mobiFLOW capture screen 43

Change the dynamic capture colors 44

Change labels and messages 45

Change the text indicators 45

Info screen popup 46

Design or change the UI 47

hintDidChange 48

setInstructionLabelText 48
bringButtonsToForground 48
Accessibility 48
Receive mobiFLOW notifications 54
Design or change the Guidelines popup UI 54
CustomInfoScreenView.h file 55
CustomInfoScreenView.mm file 55
Split capture front and back (Check only) 57
Front capture 58
Back capture 58
Captions and messages 59
Reporting issues 61
Guidelines for successful capture 62
Contrast 62
Background homogeneity 62
Lighting 62
Shooting and rotation angles 62
Taking the picture 63
Checks only: Digital row (MICR) 63

About this guide

This guide describes the 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.

Known issues and workarounds are also described.

General notes

- This SDK is relevant only to applications running on iOS 6.0 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.
- The library crops the image using the mobiFLOW SDK 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 the Linkage section.
- An integration sample in Swift can be found in the TISSimpleDemo application.
- An integration sample in Objective-C can be found in TISShowCase application.

Project settings

Installation

Manual installation

1. Add the following frameworks to the project:
   - libz.tbd
   - libc++.tbd
   - AVFoundation.framework
   - CoreVideo.framework
   - CoreMedia.framework
   - CoreMotion.framework
   - AudioToolbox.framework
   - AssetsLibrary.framework
   - QuartzCore.framework
   - ImageIO.framework
   - Accelerate.framework
2. Add the **TISMobiFLOWWidget.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**.

CocoaPods installation

1. To integrate the mobilFLOW SDK into your Xcode project using CocoaPods, specify the following in your Podfile:

```
source 'https://tisPodsUser@bitbucket.org/mobiflow/tispodsprivatespecs.git'
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '8.0'
target '<Your Target Name>' do
        use_frameworks!

        pod 'MobiFlowSDK', '~> 5.0.0'
end
```

2. Run the following command:

```
$ pod install
```

3. Insert the password provided to you by Top Image Systems.

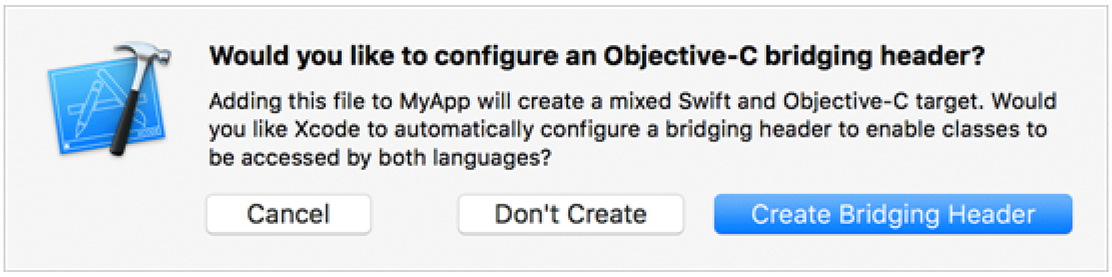
Other settings

Go to the **Summary** tab of your project. In the **General** tab, 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 <TISMobiFlowWidget/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 <TISMobiFlowWidget/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

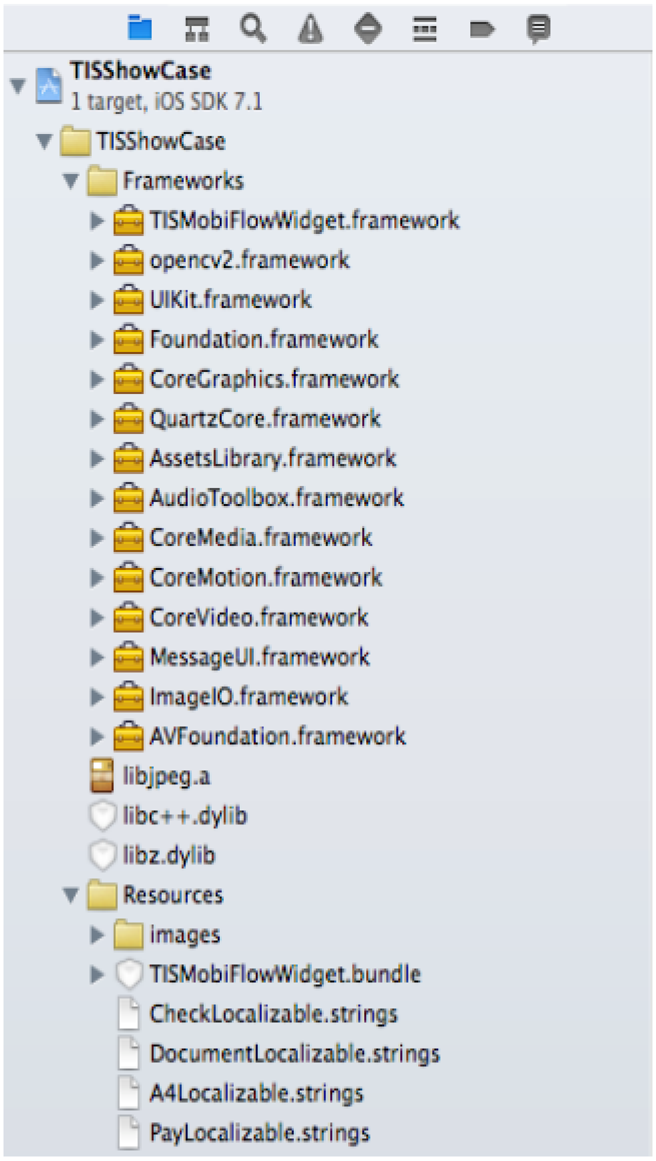
Note: If using CocoaPods, you can skip this section.

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

   Note: *TISMobiFlowWidget.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
  1 target, iOS SDK 7.1
  ▼ TISShowCase
    ▼ Frameworks
      ▶ TISMobiFlowWidget.framework
      ▶ opencv2.framework
      ▶ UIKit.framework
      ▶ Foundation.framework
      ▶ CoreGraphics.framework
      ▶ QuartzCore.framework
      ▶ AssetsLibrary.framework
      ▶ AudioToolbox.framework
      ▶ CoreMedia.framework
      ▶ CoreMotion.framework
      ▶ CoreVideo.framework
      ▶ MessageUI.framework
      ▶ ImageIO.framework
      ▶ AVFoundation.framework
      libjpeg.a
      libc++.dylib
      libz.dylib
    ▼ Resources
      ▶ images
      ▶ TISMobiFlowWidget.bundle
        CheckLocalizable.strings
        DocumentLocalizable.strings
        A4Localizable.strings
        PayLocalizable.strings
```

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*.

To initialize *TISCaptureManagerViewController* there are 2 options:

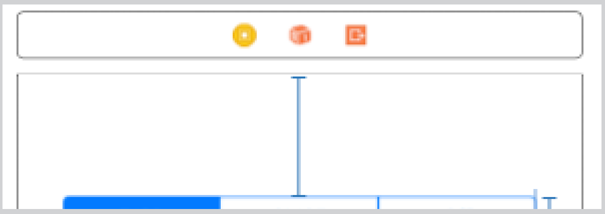
```
(nullable instancetype) initWithSessionParameters:(nonnull TISSessionParameters*)sessionParameters
andCustomView:(nullable UIViewController*)customViewController;
```

or

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

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

Tip: 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

Here is a quick example of 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)

Here is an example 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 can be found in the TIS mobiFLOW ShowCase app sample, which is included in the SDK *Bundle* package.

Session parameters

The parameters for *TISSessionParameters* should be set according to this table.

Parameter	Description
documentType	Document type set to one of the enums: ▪ TISDocumentTypeCheck ▪ TISDocumentTypeBillPayment ▪ TISDocumentTypeFullPage ▪ TISDocumentTypePassport ▪ TISDocumentTypeCard ▪ TISDocumentTypeCustom **Note:** There no default. This must be set!
debugMode	In debug mode, images are stored on the device and logs are written to the console. Default value: NO
uxType	Static capture sets predefined boundaries on the screen according to the aspect ratio, while the document must be placed relatively within the shown boundaries. Live capture looks for a quadrilateral of a document in any size, with optional additional settings according to 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* will skip validation in live mode and let you capture any document. *uxType* can be set to one of the following enums: ▪ TISFlowUXTypeStatic ▪ TISFlowUXTypeLive Default value: TISFlowUXTypeLive

Parameter	Description
minHeightWidthAspectRatio	Minimum allowed ratio between the captured image’s height and width. Default values: 0.35 (check, bill) 1.35 (full page) 0.65 (passport) 0.582 (card)
maxHeightWidthAspectRatio	Maximum allowed ratio between the captured image's height and width. Default values: 0.50 (check, bill) 1.45 (full page) 0.8 (passport) 0.7117 (card)
enableIQA	Enable or disable the IQA validations. Default value: NO
IQASettings	A class of type *TISCheckIqaParameters* to set all the threshold parameters for the IQA validations.
showInfoScreen	Show the information screen if the user has difficulty capturing the document after a specific set time. Default value: YES
InfoScreenInterval	The number of seconds until the information screen appears on the camera overlay. Default Value: 10.0

Parameter	Description
showGuidelinesIndicators	When set to NO, only two static indicators will be presented. ■ *TISFlowIndicatorAlign* – static 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 will be presented. Default value: YES
outputGrayscaleImage	Enable the output of a grayscale JPG. Default value: YES
grayscaleImageCompression	A value of the factor by which the JPG cropped grayscale image will be compressed. The value ranges from 0.0 for highest compression (lowest quality) to 1.0 (highest quality). Default value: 1.0
outputOriginalImage	Enable the output of the captured original image. Default value: YES
outputColorImage	Enable the output of the captured cropped color image. Default value: YES
colorImageCompression	A value of the factor by which the JPG cropped color image will be compressed. Values ranges from 0.0 for highest compression (lowest quality) to 1.0 (highest quality). Default value: 1.0
outputBinarizedImage	Enable the output of the captured black & white image. Default value: YES
grayScaleSize	Set the width and height of the grayscale output image. The parameter is of type *CGSize*. Default value: {0,0}

Parameter	Description
enableBlurDetection	When set to YES, mobiFLOW will check the sharpness of an image and will notify when the image is blurred. **Note:** Currently blur detection does not apply on the back side of a document. Default value: NO
videoFeedProcessing	When set to YES, the picture will be taken directly from the video feed when the document is aligned properly with the frame. In this case, the device will not switch to still mode and will not present the countdown sequence. When set to NO, the device will switch to still mode to take the picture. Must be set to YES for Passport. Default value: YES for Check and Passport, NO for other types
maxVideoFrameToCapture	When video feed processing is enabled, the library will try to process the image that was captured. 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. This parameter should be set between 5 and 10 for better performance. This parameter is relevant only when *videoFeedProcessing* is set to YES. Default value: 7
showCountDown	Applicable for still mode only. 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 user sees the *hold still* message on the screen. Default value: NO

Parameter	Description
countDownStartValue	When the counter for taking a still image is shown, it will start from the number set in this parameter. Default value: 2
countDownStopValue	When the counter for taking a still image is shown, it will count down to the number set in this parameter. Must be a number lower than *countDownStartValue*. Default value: 0
enableCountdownSound	Enable a sound along with the image capture countdown. The sound to be played is *beep.aif* from the bundle. Default value: 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 value: Nil
showDefaultProcessingView	Shows the TIS processing screen (red spinner). If set to NO, you must implement a custom processing screen using mobiFLOW notifications (see Receive mobiFLOW notifications). Default value: YES
surroundingColorForDocumentFrame	The color surrounding the document capture frame. Default Value: [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 value: YES

Parameter	Description
multiPageCapture	When set to YES, this parameter enables capture of multiple documents. After each capture, a prompt screen is displayed asking the user if they 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 value: NO
binarizeBackSameAsFront	When set to YES, the same binarization algorithm that runs on the front side will run on the back side of the check. Default value: NO
binarizationThreshold	Threshold for the strength of the binarization algorithm. Values can be between 0.0 and 1.0. This should be set only when capturing a single size document. If the size varies, like Bills, then it should be set to 0.0 for optimization. 1.0 – darkest 0.0 – The SDK calculates the optimal threshold according to the image size. Default value: 0.0
scanFrontOnly	When set to YES, only the front side will be captured. When set to NO and *scanBackOnly* is set to NO, both front and back will be captured. Default value: NO for Checks, YES for other types
scanBackOnly	When set to YES, only the back side is captured. If *scanFrontOnly* is also YES, it will fail to initialize the Library. Default value: NO

Parameter	Description
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 will be straightened and aligned from the angled position to a flat position. Using this method may impact the quality of the final image. Default value: NO
scanBarcodeLocation	Specify whether to scan the barcode in addition to the document capture session. Specify the side of the document from which capture the barcode. ■ TISScanBarcodeFront ■ TISScanBarcodeBack ■ TISScanBarcodeFrontAndBack ■ TISScanBarcodeNone Default value: TISScanBarcodeNone

Parameter	Description
barcodeTypes	Relevant only when *scanBarcodeLocation* is not *TISScanBarcodeNone*. Contains the barcode types that will be recognized during the barcode scan session. Once a barcode is detected, if there is a match with one of the barcode types, the barcode is parsed and the SDK continues to capture the document. If one of the barcode types includes *TISBarcodeTypeQRCode*, *TISBarcodeTypeAztecCode* or *TISBarcodeTypeDataMatrixCode*, a square will appear instead of a rectangle for the barcode detection. Supported barcode types in the array: ■ TISBarcodeTypeUPCECode ■ TISBarcodeTypeCode39Code ■ TISBarcodeTypeCode39Mod43Code ■ TISBarcodeTypeEAN13Code ■ TISBarcodeTypeEAN8Code ■ TISBarcodeTypeCode93Code ■ TISBarcodeTypeCode128Code ■ TISBarcodeTypePDF417Code ■ TISBarcodeTypeQRCode ■ TISBarcodeTypeAztecCode ■ TISBarcodeTypeInterleaved2of5Code ■ TISBarcodeTypeITF14Code ■ TISBarcodeTypeDataMatrixCode Default value: all barcode types

Parameter	Description
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 value: OCRType_OFF
minMICRLength (Check document type only)	Minimum MICR length (number of characters). Default value: 15
maxMICRLength (Check document type only)	Maximum MICR length (number of characters). Default value: 50
frontImageSize (Check document type only)	Size of the front black & 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.
portraitCapture (Custom document type only)	When set to YES, the camera opens in portrait mode. Default value: 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 4 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) will 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 value: (0.0, 0.0, 1.0, 1.0) – use full screen.

Parameter	Description
binarizationType (Custom Document Type Only)	Available only for the Custom document type. *TISGeneralBinarization*: Default value for all document types except Check. *TISCheckBinarization*: Default value for Check.
license	Of the type *TISLicenseParameters* class, which includes 3 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 about these parameters.
animateTransitionInLivePreview	For *TISFlowUXType.LIVE*. When set to TRUE, the green and red rectangles will switch with smooth transition animation. Default Value: YES (BOOL)
softCaptureThreshold	When enabled, the calling app will have 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 (the default value is 0, the same threshold as previous versions). A higher value will make the capture experience less strict. **Note:** At the maximum threshold, capture at a wide angle may effect image quality. Default value: 0 (float)
tapToFocus	When set to TRUE, the phone will explicitly order the device to focus on the image after the user taps on the camera overlay. Default value: YES (BOOL)
enableManualCapture	When set to TRUE, a button will be added to the screen, allowing to take a still image immediately that will be sent to processing or to the Crop Controller. Default value: NO (BOOL)

Parameter	Description
enableCropController	When set to TRUE, the image that was taken by manual capture, or automatically by the SDK, is sent to a Crop Controller to confirm the quality and cropping of the image, or to correct the cropping, before it is sent to processing. Default value: NO (BOOL)
shouldDismissWithAnimation	When set to TRUE, the capture screen will be dismissed with animation. Default value: YES (BOOL)
showErrorSignatureOverCMC7 (Check document type only)	When set to TRUE, if a signature is detected over an CMC7 MICR, an error is sent to the calling app. Default value: 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, so upgrading to a newer version requires a new license that matches the version of the SDK used.

The following 3 values (which are provided by TIS) must be initialized:

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.

Sample code:

```
sessionParameters.licenseParams = [[TISLicenseParameters alloc] initWithLicensee:@"ABCD"
licenseKey:@"a70e52b0-e499-3562-afb1-17f04038356b"
activeLicense:@"TqeRDhExXuGCLNdIcvb4OR9+QJYiTnWQ3ooFtcWx39OkkNeUYf4Ph0U+P5x6DaRIdA84HwIW
UzF5YMLA5k=="];
```

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

If the license validation failed, 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.

The parameters for *iQAParameters* should be set according to this table.

Parameter	**Description**	**Default value**
RotationSkew	Maximum skewing angle allowed.	7.5f
minDarknessFront	Minimum ratio of black pixels to total pixels for the front side.	0.009f
maxDarknessFront	Maximum ratio of black pixels to total pixels for the front side.	0.9f
minDarknessBack	Minimum ratio of black pixels to total pixels for back side.	0.0038f
maxDarknessBack	Maximum ratio of black pixels to total pixels for back side.	0.98f
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.	5852

Parameter	Description	Default value
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.	5852
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.	1.0f 1.0f 0.4f 1.0f 1.0f 0.4f 1.0f 1.0f 0.4f 0.8f 0.8f 0.3f
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.	0.3f 0.3f 0.1f 0.3f 0.3f 0.1f 0.8f 0.8f 0.3f 0.3f 0.3f 0.1f
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.	

Parameter	**Description**	**Default value**
MinImageFileSizeFront	The minimum file size for the TIFF image for the front side.	0.5f
MaxImageFileSizeFront	The maximum file size for the TIFF image for the front side.	200f
MinImageFileSizeBack	The minimum file size for the TIFF image for the back side.	0.5f
MaxImageFileSizeBack	The maximum file size for the TIFF image for the back side.	200f
horizontalStreakSumOfBlackPixels	The minimum number of black pixels required to determine if the line is black (check front).	30
horizontalStreakLineWidth	The minimum width of the black line to detect (check front).	18
horizontalStreakNumLines	The minimum number of black lines for the horizontal streaks alert (check front).	4
carbonStripSumOfBlackPixels	The minimum number of black pixels required to determine if the line is black (check back).	25
carbonStripLineWidth	The minimum width of the black line to detect (check back).	12
carbonStripNumLines	The minimum number of black lines for the horizontal streaks alert (check back).	1
piggyBackMaxWidth	Maximum width threshold between two checks that overlap each other.	10
piggyBackMaxHeight	Maximum height threshold between two checks that overlap each other.	
piggyBackMaxAR	Top and bottom Location threshold between two checks that overlap each other.	20

Parameter	Description	Default value
piggyBackMinAR	Minimum aspect ratio between two checks that overlap each other.	

For default values, see IQA Settings in the Android SDK documentation.

Leveler parameters

The parameters for *LevelerParameters* should be set 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.

Parameter	Description
alignmentToFrame	Relevant for *twoUnitLeveler* and *scaleLeveler* only. The alignment of the two leveler units to the capturing frame: ▪ topLeft ▪ bottomLeft ▪ topRight ▪ bottomRight Deafult 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
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

Parameter	Description
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

Handling messages, errors and results

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

Result delegate

The signature of *TISMobiFlowDelegate*.

In case of success capture:

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

In case of failure/cancel:

```
-(void) cancel:(nonnull TISCaptureManagerViewController*) captureManagerViewController;
```

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.

Parameter	Description
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.
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, barcode parsed string for Front, TISBarcodeType Back, barcode parsed string for Back If *scanBarcodeLocation* is set to *TISScanBarcodeNone* an empty dictionary is returned.

For document type Card only

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

Parameter	Description
panCardResultsByField	An array with the card's results. The array size changes dynamically according to the number of results found. The results are ordered by the x, y location of each field, from top to bottom and left to right.

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 barcode results of a US/Canadian driver's license.

The dictionary includes the following keys:

First Name	Date Of Birth	Audit Information
Middle Name	Sex	Ethnicity
Last Name	Issue Date	Compliance Type
Name Suffix	Restriction Code	First Name Truncation
Address	Endorsement Code	Middle Name Truncation
City	Limited Duration Document Indicator	Last Name Truncation
State	Document Number	Federal Commercial Vehicle Code
Postal Code	Country ID	Customer Specific Control Number
ID Number	Inventor Control Number	WA Specific Endorsements
Class	Card Revision Date	Transaction Types
Height	Temp Visitor	Under 18 Until
Weight	Address	Under 21 Until
Eye Color	Address Additional info	Revision Date
Hair Color	Duplicates	Social Security Number
Expiration Date	Organ Donor	

To retrieve the dictionary, use the following methods:

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

or:

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

The keys can be as follows:

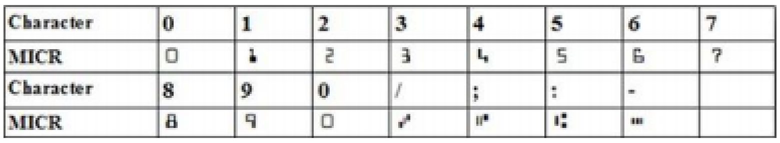
Parameter	**Description**
BARCODE_TYPE_FRONT	The type of barcode capture for the front side
BARCODE_TYPE_BACK	BACK The type of barcode capture for the back side
BARCODE_DATA_FRONT	The raw data of the barcode for the front side
BARCODE_DATA_BACK	The raw data of the barcode for the back side
BARCODE_PARSED_DATA_FRONT	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).

Parameter	Description
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:



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	⑇	⑈	⑆	⑉	

Parameter	Description
resultsScores	The score for each one of the recognized characters separated by commas, respective to the *rawResult*.

```
- (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 will always be extracted from the credit card.
expiryMonth	The expiry month of the credit card. This value will not always be found; if it is not found, the value will be 0.
expiryYear	The expiry year of the credit card. This value will not always be found; 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:

```
-(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

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 TIS mobiFLOW ShowCase app sample, which is included in the SDK *Bundle* package.

captureDidFail

The signature of *captureDidFail* is:

```
- (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 code demonstrates 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 TIS 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	TISFlowErrorImage Contrast	TISFlowErrorImage Contrast	NO*	YES

Validation description	Validation error code (enum)	Error message name	Display message on video feed processing	Message on stills
Blur Detection**	TISFlowErrorBlur Detected	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	TISFlowDigitalRow NotInScope	YES	YES
MICR Line Interruption By Signature.CMC7 Only***	TISFlowWarningMicr Interrupted	TISFlowWarningMicr Interrupted	YES	YES
IQA Folded Corner***	TISFlowErrorIQA CornerData	TISFlowErrorIQA CornerData	YES	YES
IQA Folded Edge***	TISFlowErrorIQA EdgeData	TISFlowErrorIQA EdgeData	YES	YES
IQA Skew***	TISFlowErrorIQASkew	TISFlowErrorIQASkew	YES	YES
IQA Darkness***	TISFlowErrorIQADarkness	TISFlowErrorIQA Darkness	YES	YES

Validation description	Validation error code (enum)	Error message name	Display message on video feed processing	Message on stills
IQA Number of Spots***	TISFlowErrorIQA NumSpots	TISFlowErrorIQA NumSpots	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 will proceed to process the next frame.

** Enabled on documents without OCR.

*** Checks only.

Note: IQA validations are performed only for Checks and on B&W images.

finishedMultiPageCaptureSession

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

This delegate method is called when the 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 TIS 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 TIS 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 TIS 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:

- 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.
- Upon 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, the output settings should be set 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.

Set up a custom capture user interface

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

There are 2 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 just need to change some files in the *resources\TISMobiFlowWidget.bundle* folder.

To change icons in the capture screen:

1. Find the *TISMobiFlowWidget.bundle* file in Finder and remove the *.bundle* suffix.
2. Open the *TISMobiFlowWidget* directory, and replace the desired file.
3. Add the *.bundle* suffix to the *TISMobiFlowWidget* 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 parameter with the desired colors.

The array app should fill this array in this specific order with 4 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 barcode with Static capture, when barcode 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

String name	Description
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 barcode
TISFlowPleaseCaptureCreditCard	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.

The text can be changed 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

String name	Description
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 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.

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 <TISMobiFlowWidget/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*).

Here is a short 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 TIS 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 <*TISMobiFlowWidget/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, for example, *[super initWithFrame:frame]*.
5. Then include your implementation.

The mobiFLOW library has a few UI controls where the controls' properties 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 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.

If you choose to hide the original checkBox button (which is not recommended), you must implement a call to the mobfFLOW *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.

A sample code for creating such class can be:

CustomInfoScreenView.h file

```
#import <UIKit/UIKit.h>
#import <TISMobiFlowWidget/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 TIS mobiFLOW ShowCase app sample, which is included in the SDK *Bundle* package.

In order for the mobiFLOW library to use your custom view:

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 TIS 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 TIS 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:

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 barcode in Static capture, when barcode 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
TISFlowErrorIQACornerData	Message when one of the corners of the check is missing and over the accepted threshold

Message Name	Description
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
TISFlowErrorUnknown	Message about IQA validation failure, issued if a more specific message does not apply General message about failure to validate the image, 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

Message Name	Description
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 TIS, you must reproduce the issue on the mobiFLOW Showcase app, setting debug mode to ON.

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

To be able to access these images and logs, you need a program on your computer that is able to 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.

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

<user applications>/TISShowcase/Documents/DEBUG

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:

- 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, 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" (15cm) 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 a very poor B&W image of the part that was not in the shade.

Shooting and rotation angles

The phone's camera should be lying 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 frame that is displayed, as close as possible to the frame sides.

Taking the picture

When the *hold still* wording appears, stand still with the device over the document until the countdown is over and the still picture is taken. Moving or shaking during this process can result in a blurry image, which will lead to a failure or an unclear B&W 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.