

# Kofax mobiFlow

## Android Developer's Guide

Version: 6.2.0

Date: 2024-02-21

**TUNGSTEN**  
**AUTOMATION**  
FORMERLY KOFAX

© 2012–2024 Tungsten Automation. All rights reserved.

Tungsten and Tungsten Automation are trademarks of Tungsten Automation Corporation, registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Tungsten Automation.

# Table of Contents

<b>Preface</b> .....	<b>5</b>
Product documentation.....	5
Full documentation set.....	5
Offline documentation.....	6
Getting help with Kofax products.....	6
Training.....	7
<b>Chapter 1: Project settings</b> .....	<b>8</b>
Supported architectures.....	8
Libraries.....	8
Android Studio.....	8
Android SDK version.....	10
Permissions.....	10
Features.....	10
Activities.....	10
Manifest.....	11
<b>Chapter 2: Library</b> .....	<b>12</b>
Interaction with the Library.....	12
Session parameters.....	12
Initiating image capturing.....	20
Android M permissions.....	20
License parameters.....	21
IQA parameters.....	22
Debug parameters.....	25
Document types.....	25
Video processing guidelines.....	26
Force still capture.....	26
Capture messages and errors.....	26
Session events.....	27
Error messages.....	27
UI events.....	29
Handle session continuation.....	29
Handle session results.....	32
Library flow.....	37
Clear temporary files.....	38

Security recommendations.....	38
<b>Chapter 3: Set up a custom capture user interface.....</b>	<b>40</b>
Change the look and feel.....	40
Change icons and captions.....	41
Change the text indicators.....	42
Change countdown image view.....	43
DebugRectView.....	43
Camera overlay color.....	43
Rectangle check frame.....	44
Custom view.....	44
Guidelines popup.....	44
Additional functionality in Custom view.....	44
Info screen configuration.....	45
Start a new activity that is not part of the Library.....	45
Leveler configuration.....	46
Configure levelerUI.....	46
OneUnitLeveler.....	46
Leveler parameters.....	46
TwoUnitsLeveler and Scale Leveler.....	47
Captions and messages.....	47
<b>Chapter 4: Reporting issues.....</b>	<b>50</b>
<b>Chapter 5: Guidelines for successful capture.....</b>	<b>52</b>
Contrast.....	52
Background homogeneity.....	52
Lighting.....	52
Shooting and rotation angles.....	52
Taking the picture.....	53
Digital row (MICR): Checks only.....	53
Closing camera.....	53

# Preface

This guide describes how to use the mobiFlow image capture library to integrate mobiFlow into other Android apps using Java.



- The minimum supported SDK version is 26.
- The minimum supported JDK version is 11.
- The easiest way to provide image detection in the application is to add a library project named mobiFlow. The image capturing session can be called via Intent. The mobiFlow library handles all aspects associated with the camera (integrating with the MICR algorithm for checks) and then provides the result in the Intent result.
- Image boundaries detection and contrast verification are performed on the image preview frames.
- Before sending the image to the server, the image is cropped, binarized (with 1 channel) from a color image to a B&W image, and then set to TIFF with Group 4 Fax Encoding (CCITT T.6).

## Product documentation

To access the full Kofax mobiFlow documentation set online, see the [Kofax mobiFlow Product Documentation page](#). If the security policy for your organization restricts Internet access, you can access the Kofax mobiFlow documentation offline. See [Offline mode](#).

### Full documentation set

The complete set of Kofax mobiFlow documentation is included in the following table.

Document	Description
Android Developer's Guide	Provides details about supported architectures, libraries, reporting issues, how to set up a custom capture user interface, and guidelines for successful capture.
iOS Developer's Guide	Provides information on project settings, use of various parameters of camera capture flow, and how to custom capture user interface setup.
Release Notes	Details the list of newly added features, resolved issues, known issues with applicable workaround, and the changes done since the previous release.

Document	Description
SDK Developer's Guide	Provides information on mobile capture parameters, licensing, collecting results, events and errors, look and feel, and guidelines for successful capture.
Technical Specifications	Describes the prerequisites including hardware, software, and third-party technologies to ensure proper functionality of the Kofax mobiFlow application.

## Offline documentation


To make the documentation available for use in offline mode, obtain the documentation files from the Kofax mobiFlow product package that you downloaded from the [Kofax Fulfillment Site](#). The file you need is KofaxmobiFlowDocumentation\_6.2.0\_EN.zip.

1. Extract the contents of KofaxmobiFlowDocumentation\_6.2.0\_EN.zip to a folder on your computer.
2. Copy the folder to a location that is accessible while you are working with Kofax mobiFlow.

## Getting help with Kofax products

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

To access the Kofax Knowledge Portal, go to <https://knowledge.kofax.com>.

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

The Kofax Knowledge Portal 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.  
To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or click the View All Products button.

From the Knowledge Portal home page, you can:

- Access the Kofax Community (for all customers).  
On the Resources menu, click the **Community** link.
- Access the Kofax Customer Portal (for eligible customers).  
Go to the [Support Portal Information](#) page and click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).  
Go to the [Support Portal Information](#) page and click **Log in to the Partner Portal**.

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

Go to the [Support Details](#) page and select the appropriate article.

## Training

Kofax offers both classroom and online training to help you make the most of your product. To learn more about training courses and schedules, visit the [Kofax Education Portal](#) on the Kofax website.


## Chapter 1

# Project settings

This chapter describes the project settings for the Android SDK.

### AndroidX:

The mobiFlow 6.2.0 SDK is upgraded to AndroidX. Thus, applications using this version of SDK should be updated to AndroidX. Refer to the mobiShowCase application for details on how to integrate this SDK with AndroidX supported application.

 The mobiShowCase application is provided with the SDK.

## Supported architectures

The mobiFlow SDK supports the following architectures:

- armeabi-v7a
- arm64-v8a

To reduce the APK size to a minimum, we recommend splitting the APK into multiple ABIs per architecture and uploading several APK files per architecture to Google Play.

## Libraries

### Android Studio

The project can be integrated manually using the mobiFlow library or aar. For static UI customization via XML, use the mobiFlow library integration.

#### **mobiFlow library integration**

Make sure your project contains mobiFlow module, and the Build.gradle file of the calling app contains the mobiFlow library.

1. Create a new folder with the name **libs** under your application folder.
2. Move the MobiFlow.jar file from the `mobiFlow/libs` folder to `{ApplicationFolder}/libs` folder.
3. To compile the jar file, add the following code in the dependencies section of the application build.gradle file.



```
dependencies {
    repositories {
        flatDir {
            dirs 'libs'
        }
    }
}
```

**i** The above three steps are also applicable to the mobiShowCase application.

4. Include a dependency for the mobiFLOW library in the Build.gradle file as shown below.

```
dependencies {
    implementation fileTree(dir: 'libs', include: '*.jar')
    implementation project(':mobiFLOW')
}
```

**i** You can use the configuration from the mobiShowCase application as a reference and modify it to suit your requirement.

### aar integration

1. Copy the aar file to the libs folder of the calling application.
2. To compile the aar file, add the following code in the dependencies section of the build.gradle file.

```
dependencies {
    repositories {
        flatDir {
            dirs 'libs'
        }
    }
    implementation (name: 'mobiFlow.Android-release', ext: 'aar')
}
```

**i** The Android SDK version, permissions, and features to the calling application Manifest.

Add the following additional changes to the build.gradle file.

1. Use gradle plugin 7.0.4.

```
classpath 'com.android.tools.build:gradle:7.0.4'
```

2. Add the following section under the android block.

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
```

3. Add the following dependencies to the dependencies block.

```
def camerax_version = "1.2.0-alpha02"
implementation "androidx.camera:camera-core:${camerax_version}"
implementation "androidx.camera:camera-camera2:${camerax_version}"
implementation "androidx.camera:camera-lifecycle:${camerax_version}"
```

```
implementation "androidx.camera:camera-view:${camerax_version}"
```

## Android SDK version

```
<uses-sdk android:minSdkVersion="26"/>
```

## Permissions

```
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.FLASHLIGHT" />  
<uses-permission android:name="android.permission.HIGH_SAMPLING_RATE_SENSORS"/>
```

If login information is required for the application, add the following permissions:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"  
    android:maxSdkVersion="28"/>  
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
```

## Features

```
<uses-feature android:name="android.hardware.camera" android:required="true" />  
<uses-feature android:name="android.hardware.camera.autofocus" />  
<uses-feature android:name="android.hardware.camera.flash" android:required="true" />
```

## Activities

Add full class paths to camera-related activities and instruction activities from the library project.

```
<activity  
    android:name=  
    "com.topimagesystems.controllers.imageanalyze.DynamicCaptureCameraController"  
    android:configChanges="keyboardHidden|orientation|screenSize"  
    android:hardwareAccelerated="false" />  
<!-- camera -->  
<activity  
    android:name="com.topimagesystems.controllers.imageanalyze.CameraManagerController"  
    android:configChanges="keyboardHidden|orientation|screenSize" />  
</activity>  
<activity  
    android:name="com.topimagesystems.controllers.imageanalyze.CameraController"  
    android:configChanges="keyboardHidden|orientation|screenSize"  
    android:hardwareAccelerated="false" />  
</activity>  
<activity  
    android:name="com.topimagesystems.ui.InfoScreenActivity"  
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />  
</activity>
```

Add the following element to the activity which includes Intent filters to support targetAPI 31 requirements.

```
android:exported = true
```

## Manifest

To support the Android OS 10 device storage requirements, add the following property to the application tag.

```
android:requestLegacyExternalStorage="true"
```

## Chapter 2

# Library

This chapter describes how to use the mobiFlow Android library.



## Interaction with the Library

Refer to the sample SDK application for a full sample code and integration details.


### Session parameters

Parameter	Default	Description
documentType	None. You must set this parameter.	Document type set to one of the enums: <ul style="list-style-type: none"><li>TISDocumentType.CHECK</li><li>TISDocumentType.PAYMENT</li></ul>
debugMode	FALSE	When set to TRUE, images are stored on the device, and logs are written to the console.
uxType	LIVE	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 of 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. TISFlowUXType can be set to one of the following values: <ul style="list-style-type: none"><li>STATIC</li><li>LIVE</li></ul>
minHeightWidthAspectRatio	Checks and bills: 0.35	The minimum ratio between the height and width of a captured image.


Parameter	Default	Description
maxHeightWidthAspectRatio	Checks and bills: 0.52	<p>The maximum ratio between the height and width of a captured image.</p> <p><b>i</b> CameraAPI2: If you experience difficulty while capturing checks that are longer in size, even when the device is held at the correct position, set the ratio to 0.5.</p>
enableIQA	FALSE	When set to TRUE, enables the IQA validations.
IQASettings		<p>A class of type IqaSettings to set all the threshold parameters for the IQA validations.</p> <p>You can leave it to defaults if not in use.</p>
showInfoScreen	TRUE	Shows the information screen if there is difficulty capturing the document after a specific set time.
InfoScreenInterval	10000	The number of milliseconds until the information screen appears on the camera overlay.
showGuidelinesIndicators	TRUE	<p>When set to FALSE, only two static indicators are presented.</p> <ul style="list-style-type: none"> <li>TISFlowIndicatorAlign: Indicator for alignment (the device should be aligned with the document)</li> <li>TISFlowIndicatorHold: Indicator for hold (the device should be held over the document)</li> </ul> <p>When set to TRUE, dynamic indicators are presented.</p>
outputGrayscaleImage	TRUE	<p>Enables the output of a grayscale JPG.</p> <p>The grayscale image contains metadata only for check images.</p>
grayscaleImageCompression	1.0	A value of the factor by which the cropped grayscale JPG image is compressed. The value ranges from 0.0 for the highest compression (lowest quality) to 1.0 (highest quality).
outputOriginalImage	TRUE	Enables the output of the captured original image.
outputColorImage	TRUE	Enables the output of the captured cropped color image.
colorImageCompression	1.0	<p>A value of the factor by which the cropped color JPG image is compressed.</p> <p>The value ranges from 0.0 for highest compression (lowest quality) to 1.0 (highest quality).</p>
outputBinarizedImage	TRUE	Enables the output of the captured black and white image.
grayScaleSize	[0,0], which means the image cannot be resized.	Set the width and height of the grayscale output image. The parameter is of type <code>int[]</code> .


Parameter	Default	Description
enableBlurDetection (Beta feature)	FALSE	<p>When set to TRUE, mobiFlow checks the sharpness of an image and notifies when the image is blurred.</p> <p> Currently, the blur detection does not apply to the back side of a document.</p> <p>Set to FALSE for Checks.</p>
videoFeedProcessing	TRUE for Check and FALSE for other types.	<p>Only applicable to devices with at least 1920*1080 video resolution.</p> <p>When set to TRUE, 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.</p> <p>When set to FALSE, the device switches to still mode to take the picture.</p> <p>For devices with video resolution lower than 1920*1080, the image is taken in stills mode, even if video feed processing is set to TRUE.</p> <p>See <a href="#">Captions and messages</a> for more details.</p> <p> When bills are captured with video mode, the recognition rate is low.</p>
maxVideoFrameToCapture	7	<p>When video feed processing is enabled and the video resolution of the device is higher than the minimum (), 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.</p> <p>For better performance, set this parameter between 5 and 10. This parameter is relevant only when videoFeedProcessing is set to TRUE.</p>
showCountDown	FALSE	<p>Only applicable to still mode.</p> <p>When set to TRUE, once the user is ready to take a picture, the frame turns green, and a countdown is shown until the picture is taken automatically.</p> <p>When set to FALSE, no countdown is shown. The picture is taken when the frame is green and the HOLD STILL message appears on the screen.</p>
countDownStartValue	2	The number from which the countdown starts when the counter for taking a still image is set in this parameter.
countDownStopValue	0	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.


Parameter	Default	Description
enableCountdownSound	FALSE	When set to TRUE, enables a sound along with the image capture countdown. The sound that is played is beep.
dynamicStrings	Null	Option to change Strings.xml values at runtime. The key is the String ID as shown in the Strings.xml, the value is the string to replace. If a key does not have a value or, all the object is null, the default value is taken from the Strings.xml (set to Null - no change from the original behavior). The parameter is of the type <code>HashMap&lt;String, String&gt;</code> .
showDefaultProcessingView	TRUE	Shows the processing screen (red spinner). When set to TRUE, the progress bar is presented over the camera overlay. When set to FALSE, a new layout from <code>mback_camera_layout</code> is presented on the screen when the processing stage has started ( <code>@+id/processingOverlay</code> ).
binarizeBackSameAsFront	FALSE	When set to TRUE, the same binarization algorithm that runs on the front side runs on the back side of the check.
binarizationThreshold	0.0	The 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.
scanFrontOnly	FALSE for Checks, TRUE for other types	When set to TRUE, only the front side is captured. When set to FALSE and <code>scanBackOnly</code> is also set to FALSE, both front and back sides are captured.
scanBackOnly	FALSE	When set to TRUE, only the back side is captured. If <code>scanFrontOnly</code> is also TRUE, the system fails to initialize the Library.
softCapture	FALSE	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.

Parameter	Default	Description
scanBarcodeLocation	BARCODE_NONE	<p>The value of type TISScanBarcodeLocation enum. When set to BARCODE_NONE, no barcode is detected. Otherwise, it determines on which side of the document the barcode is detected.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> <li>• BARCODE_FRONT</li> <li>• BARCODE_BACK</li> <li>• BARCODE_FRONT_AND_BACK</li> <li>• BARCODE_NONE</li> </ul>
barcodeTypes	All barcode types	<p>ArrayList&lt;TISBarcodeType&gt; Relevant only when scanBarcodeLocation is not NONE.</p> <p>Contains the barcode types that are recognized during the barcode scan session.</p> <p>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.</p> <p>If one of the barcode types includes QR_CODE, AZTEC_CODE, or DATA_MATRIX_CODE, a square appears instead of a rectangle for the barcode detection.</p> <p>Supported barcode types in the array:</p> <ul style="list-style-type: none"> <li>• UPCE_CODE</li> <li>• CODE_39_CODE</li> <li>• CODE_39_MOD_43_CODE</li> <li>• EAN_13_CODE</li> <li>• EAN_8_CODE</li> <li>• CODE_93_CODE</li> <li>• CODE_128_CODE</li> <li>• PDF_417_CODE</li> <li>• QR_CODE</li> <li>• AZTEC_CODE</li> <li>• INTERLEAVED_2_OF_5_CODE</li> <li>• ITF_14_CODEDATA_MATRIX_CODE</li> </ul> <div style="background-color: #e0f2f7; padding: 5px; border: 1px solid #ccc;"> <p> This feature is deprecated, and it will be removed in a future release.</p> </div>
customView	FALSE	When set to TRUE, uses the custom view that was set up.



Parameter	Default	Description
ocrType	OFF	<ul style="list-style-type: none"> <li>UNKNOWN (For Check only)</li> <li>E13B (For Check only)</li> <li>CMC7 (For Check only)</li> <li>OFF</li> </ul>
useMaxResolution	FALSE	<p>This setting is valid for the still capture mode only. The SDK uses the optimal resolution to process an image automatically per document type and flow. Set this parameter to TRUE, if you need to use the maximum camera resolution of the device.</p> <div style="border: 1px solid #add8e6; padding: 5px; background-color: #e6f2ff;"> <p> Setting this parameter to TRUE can lead to a longer processing time.</p> </div>
minMICRLength (Check capture only)	15	Minimum MICR length (number of characters).
maxMICRLength (Check capture only)	50	Maximum MICR length (number of characters).
frontImageSize (Check capture only)	Null	<p>Size of the front black and white and grayscale images output.</p> <p>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.</p> <p>The first value in the array is the image width and the second is the image height. Parameter type is Int[].</p>
binarizationType (All document types except Check)	<p>TIS_GENERAL_BINARIZATION: The default value for all document types except Check.</p> <p>TIS_CHECK_BINARIZATION: The default value for Check.</p>	<p>TISSauvolaBinarization</p> <p>TISOtsuAdaptiveBinarization</p>
license		<p>Of the type TISLicenseParameters class, which includes three members that must be initialized on the constructor.</p> <p>A valid license must be coded for the camera session to start; otherwise, a license error message is displayed.</p> <p>See <a href="#">License parameters</a> for more information.</p>
animateTransitionInLiveP review	TRUE	For TISFlowUXType.LIVE. When set to TRUE, the green and red rectangles switch with a smooth transition animation.

Parameter	Default	Description
softCaptureThreshold	0 (the same threshold as previous versions).	<p>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.</p> <p>Possible values are 0–1.</p> <p>A higher value makes the capture experience less strict.</p> <div style="background-color: #e0f2f7; padding: 5px; border: 1px solid #ccc;"> <p> At the maximum threshold, capture at a wide angle may affect image quality.</p> </div>
tapToFocus	TRUE	When set to TRUE, allows you to tap on the camera overlay to focus explicitly.
enableBlurDetectionOnBackSide (Beta feature)	FALSE	When set to TRUE, the SDK performs blur detection on the back side of the document to avoid blurry back side images. Valid for all document types.
enableManualCapture	FALSE	When set to TRUE, a button is added to the screen, allows you to take a still image immediately that will be sent to processing or the Crop Controller.
enableCropController	FALSE	<p>When set to TRUE, 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.</p> <p>This feature is only available from Android API 11.</p>
shouldDismissWithAnimation	TRUE	Dismisses the capture screen with animation.
showErrorSignatureOverCMC7	TRUE	When set to TRUE, if a signature is detected over a CMC7 MICR, sends an error to the calling app.
showGridInLivePreview	TRUE	If the uxType parameter is LIVE, displays the grid over the camera overlay.
maxFrameWidth	Default value for both landscape and portrait: 0.99	If the uxType is STATIC, determines the percentage of the screen width used by the capture frame.
minFrameHeight	Default value for landscape: 0.75 Default value for portrait: 0.86	If the uxType is STATIC, determines the percentage of the screen height used by the capture frame.

Parameter	Default	Description
searchForSignature (Check document type only)	SIGNATURE_NONE	<p>Verifies whether there is a signature on the check. Of the TISearchForSignature enum value. When set to SIGNATURE_NONE, no signature is searched for. Otherwise, the side of the document where the signature should be found is determined.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> <li>• SIGNATURE_FRONT</li> <li>• SIGNATURE_BACK</li> <li>• SIGNATURE_FRONT_AND_BACK</li> <li>• SIGNATURE_NONE</li> </ul>
enableTorch	FALSE	When set to TRUE, the flashlight is enabled by default.
showInfoScreenMultipleTimes	FALSE	<p>When set to TRUE, the information screen appears multiple times.</p> <p>The showInfoScreenMultipleTimes parameter takes priority if both showInfoScreen and showInfoScreenMultipleTimes are set to TRUE. To hide the information screen, set both showInfoScreen and showInfoScreenMultipleTimes to false.</p>
enableAutoCaptureInManualMode	FALSE	When the ManualCapture mode is enabled, this parameter launches the Auto Capture Experience by default. The Manual Capture button is hidden or visible based on the toggle state of the Auto Capture.
secureCaptureScreen	FALSE	When set to TRUE, does not allow you to take screenshots of the capture screen.
useCameraXAPI	FALSE (camera will be launched).	<p>When set to TRUE, CameraX is launched from the mobiFlow SDK.</p> <div style="background-color: #e0f2f7; padding: 10px; border: 1px solid #ccc;"> <p> This feature is applicable only for the check and bill component.</p> <ul style="list-style-type: none"> <li>• CameraX is a Beta feature from Google, hence this may not work as expected.</li> <li>• This feature is deprecated, and it will be removed in a future release.</li> </ul> </div>
useCameraAPI2	FALSE (camera will be launched).	When set to TRUE, Camera2 is launched from the mobiFlow SDK.
validRectPaintStroke	0xFF82BE37 (Medium Green)	Stroke color for a valid rectangle frame.
invalidRectPaintStroke	0xFFD60A16 (Dark Red)	Stroke color for an invalid rectangle frame.
validRectPaintFill	0x60B9DE93 (Soft Aqua)	Fill color for a valid rectangle frame.

Parameter	Default	Description
invalidRectPaintFill	0x00000000 (Transparent)	Fill color for an invalid rectangle frame.

## Initiating image capturing

To initiate image capturing, invoke an intent that will trigger the library code. Consider using a convenience class `CaptureIntent` that wraps intent invoking and result receiving.

Create a default parameter class and change any parameter if needed. This example is for bill payment. See [Document types](#) for information on the different types of document.

### Payments

```
CaptureIntent captureIntent = new CaptureIntent(activity);
// create CaptureIntent to interact with the library
paymentCaptureParams input = (paymentCaptureParams)
CaptureIntent.getCaptureParams(TISDocumentType.PAYMENT);
// Create parameters class that matches the document type capture using casting
captureIntent.captureDocument(input, mActivityResultLauncher);
```

### Checks

```
CaptureIntent captureIntent = new CaptureIntent(this); // this is your Activity context
IQASettingsIntent iqaSettings = IQASettingsIntent iqaSettings = new
IQASettingsIntent().getIQASettingsDefault();// init default IQA settings class
// check 51 init
// IQASettingsIntent iqaSettings = new IQASettingsIntent();
//iqaSettings = iqaSettings.getIQASStandart51Defaults();
checkCaptureParams input = (checkCaptureParams)
captureIntent.getCaptureParams(TISDocumentType.CHECK); // init default check
configuration.
input.outputColorImage = false; // cropped color image, seems you don't need this from
the settings.
input.IQASettings = iqaSettings;
input.uxType = TISFlowUXType.STATIC;
input.license = new TISLicenseParameters(... , ... , ) // input the license here
CameraController.registerListener(this); // this == class context, register listener to
get messages from the SDK, does not have to be here, can be anywhere before .
captureIntent.captureDocument(input, mActivityResultLauncher); // open camera screen.

CaptureIntent.getCaptureParams(TISDocumentType.CHECK);
```

1. Initialize `CaptureIntent` to interact with the library.
2. Get the default class parameters (using casting by adding the correct enum that describes the document type).
3. Initialize the session with the parameters class.

## Android M permissions

For Android M target applications, the SDK must have camera permission to start a session. You must choose whether to ask for permission before initiating mobiFlow or let mobiFlow ask for permission.

**i** To allow the alert to show on the calling app screen, you must have the camera permission before calling mobiFlow.

The following code shows how to request for camera permission from the calling application:

```
@TargetApi(23)
private boolean askPermission(){
Context c = getApplicationContext();

String permission = "android.permission.CAMERA";
int res = this.checkCallingOrSelfPermission(permission);
if (res == PackageManager.PERMISSION_GRANTED){
// has permission start mobiFLOW
}
Else{
// don't have permission, ask for permission - result will return to
// onRequestPermissionsResult method
requestPermissions(
new String[]{Manifest.permission.CAMERA},
MY_PERMISSIONS_REQUEST_CAMERA);
}
return true;
}
// permission result callback
@Override
public void onRequestPermissionsResult(int requestCode,
String permissions[], int[] grantResults) {
switch (requestCode) {
case Constants.CAMERA_PERMISSIONS_REQUEST: {
// If request is cancelled, the result arrays are empty.
if (grantResults.length > 0
&& grantResults[0] == PackageManager.PERMISSION_GRANTED) {
// permission was granted,
// start mobiFLOW
} else {
// permission denied !
}
return;
}
}
}
```

If you use the debug mode where images are saved on the device, you must grant storage access permission as well in the same way. This must be done from the calling app.

If mobiFlow handles the camera permission, the alert is shown before the camera opens. If the user approves the permission, the session starts as usual.

If the user denies permission, the session closes with the following result code:

```
RESULT_CAMERA_PERMISSION_ACSES_DENIED
```

## 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's name, the license key, and the license itself. The license is limited by expiration date or 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 such as the customer or the project name.
licenseKey	A unique key provided to each license or customer.
activeLicense	An encrypted string that contains the license information.

Following is the sample code for license.

```
input.license = new
    TISLicenseParameters("ABCD", "a70e52b0-e499-3562-
afb1-17f04038356b", "TqeRDhExXuGCLNdIcvb4OR9+QJYiTnWQ3ooFtcWx39OkkNeUYf4Ph0U
+P5x6DaRIaA84Hw1WUzF5YMLA5k==");
```

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

If the license validation fails, a notification is displayed on the screen to the user and the session ends with the following result code:

RESULT\_LICENSE\_INVALID.

To get the message you can call the following:

```
String licenseErrorMessage = data.getStringExtra(CaptureIntent.MOBIFLOW_ERROR_DETAILS);
```

See the [Handle session results](#) section for more details.

## IQA parameters

Create the IQASettingsIntent instance and set its IQA properties.

IQA is used to define validation for image quality.

Set the parameters for iQAParameters according to the following table.

Parameter	Default	Description
maxRotationSkew	7.5	The maximum skewing angle.
minDarknessFront	0.009	The minimum ratio of black pixels to total pixels for the front side.
maxDarknessFront	0.9	The maximum ratio of black pixels to total pixels for the front side.
minDarknessBack	0.0021	The minimum ratio of black pixels to total pixels for the back side.
maxDarknessBack	0.98	The maximum ratio of black pixels to total pixels for the back side.

Parameter	Default	Description
numberOfSpotsFront	5750	The maximum number of spots per square inch on average for the front side. Black areas are counted 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	5750	The maximum number of spots per square inch on average for the back side. Black areas are counted 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 cornerDataFrontTLH cornerDataFrontTLW cornerDataFrontTLA cornerDataFrontTRH cornerDataFrontTRW cornerDataFrontTRA cornerDataFrontBLH cornerDataFrontBLW cornerDataFrontBLA cornerDataFrontBRH cornerDataFrontBRW cornerDataFrontBRA		Thresholds for height, width, and area (in inches) for every corner of the check on the front side. Use the setCornerFrontSameToAllCorners function to set the same height, width, and area for all corners, or use SetCornerFrontAll to set a different threshold for each corner.
CornerDataArrayBack cornerDataBackTLH cornerDataBackTLW cornerDataBackTLA cornerDataBackTRH cornerDataBackTRW cornerDataBackTRA cornerDataBackBLH cornerDataBackBLW cornerDataBackBLA cornerDataBackBRH cornerDataBackBRW cornerDataBackBRA		Thresholds for height, width, and area (in inches) for every corner of the check on the back side. Use the setCornerBackSameToAllCorners function to set the same height, width, and area for all corners, or use SetCornerBackAll to set a different threshold for each corner.

Parameter	Default	Description
edgeDataTH edgeDataTW edgeDataTA edgeDataRH edgeDataRW edgeDataRA edgeDataBH edgeDataBW edgeDataBA edgeDataLH edgeDataLW edgeDataLA		Thresholds for height, width, and area (in inches) for every side of the check (top/bottom/left/right). Use the setEdgeSameToAllSides function to set the same height, width, and area for all corners, or use SetEdgeAll to set a different threshold for each corner.
minImageFileSizeFront	500	The minimum file size for the TIFF image for the front side.
maxImageFileSizeFront	200000	The maximum file size for the TIFF image for the front side.
minImageFileSizeBack	500	The minimum file size for the TIFF image for the back side.
maxImageFileSizeBack	200000	The maximum file size for the TIFF image for the back side.
horizontalStreakSumOfBlackPixels	25	The minimum number of black pixels required to determine if the line is black (check front).
horizontalStreakLineWidth	12	The minimum width of the black line to detect (check front).
horizontalStreakNumLines	3	The minimum number of black lines for the horizontal streaks alert (check front).
carbonStripSumOfBlackPixels	25	The minimum number of black pixels required to determine if the line is black (check back).
carbonStripLineWidth	12	The minimum width of the black line to detect (check back).
carbonStripNumLines	1	The minimum number of black lines for the horizontal streaks alert (check back).
piggyBackMaxWidth	0.5	The maximum width threshold between two checks that overlap each other.
piggyBackMaxAR	3.1	Top and bottom location threshold between two checks that overlap each other.
maxImageDimensionsHeight		The maximum height of the image to detect.
maxImageDimensionsWidth		The maximum width of the image to detect.

The following code is an example of how to initialize IQASettingsIntent and add it to the capture parameters.

```
checkCaptureParams input = (checkCaptureParams)
```



```
CaptureIntent.getCaptureParams (TISDocumentType.CHECK);
IQASettingsIntent iqaSettings = new IQASettingsIntent();
iqaSettings.cornerDataFrontTLH = 1.0f;
iqaSettings.cornerDataFrontTLW = 1.0f;
input.IQASettings = iqaSettings; (can be found in sample SDK app).
```

Use the following code to initialize the default IQA parameters:

```
check 21
iqaSettings = new IQASettingsIntent();
iqaSettings = iqaSettings.getIQASettingsDefault();
check 51:
IQASettingsIntent iqaSettings = new IQASettingsIntent();
iqaSettings = iqaSettings.getIQASStandart51Defaults();
```

## Debug parameters

You can configure the camera with values different from the mobiFlow defaults in the integer.xml file.

The SDK chooses the video and still camera resolution that best fits the devices that were tested. For problematic devices, you can choose the resolution manually by changing these values. We recommend that you first consult Kofax before changing these values.

See the following table for the parameters and their purpose.

Purpose	Parameter
To change the video width resolution	<integer name="videoWidthResolution">0</integer>
To change the video height resolution	<integer name="videoHeightResolution">0</integer>
To change the still width resolution	<integer name="stillWidthResolution">0</integer>
To change the still height resolution	<integer name="stillHeightResolution">0</integer>

**i** These parameters are for debugging only, therefore their values must be changed only for debugging purposes. When not debugging, the values must remain zero.

## Document types

The available document types are CHECK and PAYMENT.

Initialize the following document types with the default parameters that fit the relevant document type.

### Payment capture parameters

```
paymentCaptureParams input = (paymentCaptureParams)
CaptureIntent.getCaptureParams (TISDocumentType.PAYMENT);
```

### Check capture session parameters

```
checkCaptureParams input = (checkCaptureParams)
CaptureIntent.getCaptureParams (TISDocumentType.CHECK);
```

All the default parameters for each class can be modified in the calling application by accessing each parameter in the params variable (in this example, input).

Once all are initialized, you call CaptureDocument to start the camera session:

```
captureIntent.captureDocument(input, mActivityResultLauncher);
```

## Video processing guidelines

The check capture and MICR recognition process is done directly from the video feed on supported devices to achieve a better user experience and maximize performance. The video feed is supported on devices with at least 1980\*1080. On other devices, the library starts in stills mode, even when the video mode option is enabled. The MICR recognition on these devices is done only after the still picture is taken.

To avoid a long session on problematic checks, you can also configure the maximum taken frame failure in video mode with maxVideoFramesToCapture parameters. After X failures on a video frame, the Library switches to still mode.

The resources should also be imported to the library `res` folder.

### Force still capture

There is an option to add exceptional devices that will capture in still even if they support video capture (some devices capture better in still and have issues in video capture). This will enforce the devices on the list to capture in still when videoFeedProcessing is set to TRUE.

To add exceptional devices, add an entry to exception\_devices\_name\_stills\_only in the arrays.xml file.

To add a device to the exception list, create the structure in the following format:

Build.MANUFACTURER + <space> + Build.MODEL device brand, space, device model

For example: <item>Samsung SM-G900V</item>

## Capture messages and errors

To capture messages from the library, the calling app activity should register with CameraController.registerListener(this) and implement the TISmobiFLOWMessages interface.

You must import  
com.topimagesystems.controllers.imageanalyze.CameraController.TISmobiFLOWMessages.

Three public methods are generated automatically after implementing the listener.

## Session events

Use the following method to receive the session events like OCRResult, CaptureBack, and MultiCapture:

```
public void onmobiFLOWGeneralMessageReceived (TISFlowOutputMessages message, Object[]
  extraData, Context context)
```

Possible values for TISFlowGeneralMessages

Value	Description
CAPTURE_BACK	Fires when the front capture ends before continuing to capture the back, when both front and back are captured in the same session.
CHECK_OCR_RESULT	In video mode only, fires when the document type is Check and OCRType is E13B or CMC7 after recognition is done.
BACK_PRESSED	Fires when the Back button was pressed.  This event must return a value when fired. It can use either CONTINUE_CURRENT_SESSION to cancel the back default behavior, or CONTINUE_MOBI_FLOW to apply the default back press action.
extraData	This object contains the OCR results that are returned from mobiFlow and described above.

## Validate the OCR result

When one of the OCR result messages is fired, you can validate the OCR results. See [Handle session continuation](#) for information on how to confirm or reject the OCR result.

## Error messages

Use the following method to receive error messages from the Library.

```
public void onmobiFLOWErrorMessageReceived (TISFlowErrorMessage error, Object[]
  extraData, Context context)
```

Available error codes for TISFlowErrorMessage:

```
enum TISFlowErrorCode {
  ERROR_GENERAL_FAIL (R.string.TISErrorImageGeneral),
  ERROR_OCR_READING (R.strings.TISFlowErrorReadingOCRMessage),
  ERROR_NO_VALID_BOUNDING_BOX (R.string.TISFlowErrorNoValidBoundingBox),
  ERROR_IQA_CORNER_DATA (R.string.TISFlowErrorIQACornerData),
  ERROR_IQA_EDGE_DATA (R.string.TISFlowErrorIQAEdgeData),
  ERROR_IQA_SKEW (R.string.TISFlowErrorIQASkew),
  ERROR_IQA_DARKNESS (R.string.TISFlowErrorIQADarkness),
  ERROR_IQA_NUM_SPOTS (R.string.TISFlowErrorIQANumSpots),
  ERROR_IQA_HORIZONTAL_STREAK (R.string.TISFlowErrorHorizontalStreaks),
  ERROR_IQA_CARBON_STRIP (R.string.TISFlowErrorCarbonStrip),
  ERROR_IQA_PIGGY_BACK (R.string.TISFlowErrorPiggyBack),
  ERROR_IQA_IMAGE_DIMENSIONS (R.string.TISFlowErrorMinImageDimensions),
  ERROR_BLUR_DETECTED (R.string.TISErrorBlurFail),
  ERROR_MICR_LENGTH (R.strings.TISFlowErrorReadingMessage),
  ERROR_MICR_INTERRUPTED (R.strings.TISFlowMicrInterrupted),
  ERROR_MICR_ON_BACK (R.strings.TISFlowWarningMICRDetectedOnCheckBack),
  UNSUPPORTED_CAMERA,
```

```

UNSUPPORTED_AUTO_FOCUS,
UNSUPPORTED_CPU
}

```

The order in which the validations run is different when using still mode and video mode, and so are the messages that are used. The following table shows the order of 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	TISFlowErrorNoValidBoundingBox	TISFlowErrorNoValidBoundingBox	NO*	YES
The user is capturing the front side instead of the back side of the check***	TISFlowWarningMICRDetectedOnCheckBack	TISFlowWarningMICRDetectedOnCheckBack	YES	YES
OCR Validation	TISFlowErrorOCRReadingCheck	TISFlowErrorReadingMessage	YES	YES
MICR Length Validation***	TISFlowErrorMICRLength	TISFlowDigitalRowNotInScope	YES	YES
MICR Line Interruption By Signature.CMC7 Only***	TISFlowWarningMicInterrupted	TISFlowWarningMicInterrupted	YES	YES
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***	TISFlowErrorHorizontalStreaks	TISFlowErrorHorizontalStreaks	YES	YES
IQA Carbon Strip***	TISFlowErrorCarbonStrip	TISFlowErrorCarbonStrip	YES	YES
IQA Piggy Back***	TISFlowErrorPiggybackFound	TISFlowErrorPiggyback	YES	YES

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

\*\* Enabled on documents without OCR.

\*\*\* Checks only.

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

## UI events

The following function is fired when any of the UI changes happen in the following table:

```
public void on MobiFlowUIEventMessageReceived(TISFlowUIMessages message, ViewGroup cameraOverlayView)
```

You can use `cameraOverlayView.findViewById`, to inflate your additional UI element and add functionality to it.

The message parameter indicates which UI event is currently occurring in the camera controller.

Optional values for TISFlowUIMessages:

Value	Description
BEFORE_PROCESSING	Fires before the processing view animation starts.
AFTER_PROCESSING	Fires before the processing view animation finishes (library finished processing).
INIT_LAYOUT	Fires when the screen is refreshed.
HINT_CHANGED	Fires when the capture hint is changed.
INSTRUCTION_CHANGED	Fires when the capture instruction is changed.

### Accessibility

Using HINT\_CHANGED and INSTRUCTION\_CHANGED messages, you can control the accessibility of the controls and change them at runtime. See the "Sample code" in the [Handle session continuation](#).

### Signature Events

This function will indicate that the user is continuing without a signature on the check.

```
public void continuingWithOutSignature(boolean isFrontCheque);
// isFrontCheque indicates the side of the check.
```

## Handle session continuation

This section is relevant only for general and error message handling. It explains how to instruct mobiFlow to continue the session after a message is handled.

The calling app can decide at every step how mobiFlow will proceed with the enum TISFlowInputMessages.

**i** You must return a message to mobiFlow, so if you do not have any specific request, use the default: CONTINUE\_MOBI\_FLOW.

First, get the listener by using the following:

```
returnMessage = CameraController.getManagerListener();
```

Possible outgoing values for TISFlowInputMessages to instruct mobiFlow to take certain actions.

Value	Description
CONTINUE_MOBI_FLOW	Continue the normal flow. The mobiFlow alerts are displayed. <code>returnMessage.onMessageReturn (TISFlowActionCallback.CONTINUE_MOBI_FLOW);</code>
CONTINUE_MOBI_FLOW_CUSTOM_UI	Continue the normal flow without mobiFlow alerts. The calling app will get the screen context and add UI elements to the camera overlay to be displayed, instead of showing the mobiFlow default alerts.
OCR_RESULT_FAILED	When OCR validation fails runs OCR on the next frame.
OCR_RESULT_OK	When OCR is OK, proceed with the flow to process the image.
CONTINUE_CURRENT_SESSION	Relevant for the BACK_PRESSED event only. It will not close the camera activity on BACK_PRESSED. The calling application can perform some custom actions here.
CANCEL_SESSION	Stop the current session. To fetch the image if it was successfully processed, you will be able to do so on onActivityResult in the case RESULT_CLOSE_SESSION. In code: <code>returnMessage.onMessageReturn (TISFlowActionCallback.CANCEL_SESSION);</code>

### Sample code

```
@Override
public void onmobiFLOWGeneralMessageReceived (TISFlowGeneralMessages message, Object[]
  extraData, Context
  context) {
  // get messages from the library.
  returnMessage = CameraController.getManagerListener();
  switch (message) {
  case CAPTURE_BACK:
  if (errorMessageReceived) { // if got error on front image don't
  // proceed to capture back, close
  // session with image result.
  returnMessage.onMessageReturn (TISFlowInputMessages.CONTINUE_MOBI_FLOW);
  } else {
  returnMessage.onMessageReturn (TISFlowInputMessages.CONTINUE_MOBI_FLOW);
  }
  break;

  case BACK_PRESSED:
  returnMessage.onMessageReturn (TISFlowInputMessages.CONTINUE_MOBI_FLOW);
  break;

  default: // must use default value here!
  returnMessage.onMessageReturn (TISFlowInputMessages.CONTINUE_MOBI_FLOW);
  break;
  }
}

@Override
public void onmobiFLOWErrorMessageReceived (TISFlowErrorMessage error, Object[]
  extraData, Context context) {
```

```
// get Error messages from the library.
returnMessage = CameraController.getManagerListener();
switch (error) {
case ERROR_OCR_READING:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
errorMessageReceived = true;
break;
case ERROR_IMAGE_CONTRAST:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
errorMessageReceived = true;
break;
case ERROR_MICR_LENGTH:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
errorMessageReceived = true;
break;
case ERROR_NO_VALID_BOUNDING_BOX:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_BLUR_DETECTED:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_IQA_DARKNESS:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_IQA_CORNER_DATA:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_IQA_EDGE_DATA:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_IQA_SKEW:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_IQA_NUM_SPOTS:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_MICR_INTERRUPTED:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case ERROR_MICR_ON_BACK:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case UNSUPPORTED_CAMERA:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
case UNSUPPORTED_CPU:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
default:
returnMessage.onMessageReturn(TISFlowInputMessages.CONTINUE_MOBI_FLOW);
break;
}
return;
}

// For accessibility support
private TextView hintTextView;
private TextView instructionTextView;

@Override
public void onmobiFLOWUIEventMessageReceived (TISFlowUIMessages message, ViewGroup
cameraOverlayView) {
// get UI messages from the library.
returnMessage = CameraController.getManagerListener();
```

```
switch (message) {
case INIT_LAYOUT:
if (AccessibilityUtils.isAccessibilityEnabled(this)) {
if (isDynamicCapture)
hintTextView = (TextView) cameraOverlayView.findViewById(R.id.DynamicTxtIndicator);
else if (isCustomView) {
hintTextView = (TextView) cameraOverlayView.findViewById(R.id.customTxtIndicator);
instructionTextView = (TextView) cameraOverlayView.findViewById(R.id.customTxtCapture);
} else {
hintTextView = (TextView) cameraOverlayView.findViewById(R.id.txtIndicator);
instructionTextView = (TextView) cameraOverlayView.findViewById(R.id.txtCapture);
}
}
break;
case AFTER_PROCESSING:
break;
case BEFORE_PROCESSING:
break;
case HINT_CHANGED:
if (AccessibilityUtils.isAccessibilityEnabled(this)) {
if (hintTextView != null) {
String hintText = hintTextView.getText().toString();
if (hintText.equalsIgnoreCase(getResources().getString(R.string.
TISFlowIndicatorAlignFlat)))
hintTextView.setContentDescription(getResources().
getString(R.string.TISFlowIndicatorAlignFlatDescription));
}
}
break;
case ` :
if (AccessibilityUtils.isAccessibilityEnabled(this)) {
if (instructionTextView != null)
instructionTextView.setContentDescription("instruction: " +
instructionTextView.getText());
}
break;
default:
break;
}
}
```

See the more detailed sample code in the Showcase app in the ShowCaseActivity.

## Handle session results

After CaptureIntent has finished and onActivityResult() function is called, the object that is transferred back to the original calling function is SessionResultParams (see example later on this section).

CaptureIntent.MOBI\_FLOW\_REQUEST\_CODE: This Request code contains RESULT\_OK and RESULT\_CANCELLED.

### Get the images from the library

By default, the images are not saved to the device (to do so, call saveImagesToDevice());).

All the images are stored in the device memory and can be retrieved from the following:

- SessionResultParams.tiffFront;
- SessionResultParams.jpegBWFfront;



- SessionResultParams.grayscaleFront;
- SessionResultParams.colorFront;
- SessionResultParams.originalFront;
- SessionResultParams.tiffBack;
- SessionResultParams.jpegBWBack;
- SessionResultParams.grayscaleBack;
- SessionResultParams.colorBack;
- SessionResultParams.originalBack;

The example (later in the section) demonstrates how to retrieve the result from the library. It uses the optional saveImageToDevice function which is given as well in the showcase.

For Checks, when using split capture for front and back, or for other document types, if you want to scale the back side the same as the front side, use this function to retrieve it and to set the frontImageSize for the back side capture.

Method	Description
getFrontImageRectArray()	The array contains four elements of the rectangle: <ul style="list-style-type: none"> <li>• getFrontImageArray()[0] : Top-left (x) position of the rectangle</li> <li>• getFrontImageArray()[1] : Top-left (y) position of the rectangle</li> <li>• getFrontImageArray()[2] : Width of the rectangle</li> <li>• getFrontImageArray()[3] : Height of the rectangle</li> </ul>


**For document type Check only**

Methods	Description																																				
getOcrParams(String[])	Session OCR result.																																				
getOcrParams(SessionResultParams.DIGITAL_ROW_LENGTH)	The MICR length.																																				
getOcrParams (SessionResultParams.OCR_RESULT_WITH_DELIMETER)	The MICR result formatted in mobiFlow format (special characters represented by a dash).																																				
getOcrParams(SessionResultParams.OCR_RAW_RESULT)	<p>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</p> <p>The numbers represent the MICR in the following order.</p> <table border="1"> <thead> <tr> <th>Character</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> </tr> </thead> <tbody> <tr> <td>MICR</td> <td>0</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> </tr> <tr> <th>Character</th> <th>8</th> <th>9</th> <th>0</th> <th>/</th> <th>:</th> <th>:</th> <th>-</th> <th></th> </tr> <tr> <td>MICR</td> <td>8</td> <td>9</td> <td>0</td> <td>/</td> <td>:</td> <td>:</td> <td>-</td> <td></td> </tr> </tbody> </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	/	:	:	-	
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	/	:	:	-																														
getOcrParams(SessionResultParams.SCORE_RESULT)	The score for each recognized characters separated by commas, respective to the rawResult.																																				

**For document type Check only, with CMC7 MICR**

Parameter	Description
signatureOverMicrDetected	Will be available if showErrorSignatureOverCMC7 was set to FALSE. If it returns TRUE, the SDK found a signature over the MICR line.

**Barcode configuration and barcode result**

 This feature is deprecated, and it will be removed in a future release.

The SDK searches for the barcode for “x” frames (default 13). Only frames with a valid rectangle are calculated. If a barcode was not found for "x" frames, the SDK skips the barcode recognition and only captures the image.

You can modify the number of frames by changing the value of max\_barcode\_tries variable in the integers.xml file.

You can retrieve the barcode result from the SessionResultParams object using the getBarcodeResult() function.

Method	Return
getBarcodeResult()	BarcodeResult object that holds the data of the barcode scanning.

The BarcodeResult holds the barcode scanning data, and has the following get methods.

Method	Return
isEmpty()	Boolean. TRUE if no barcode data was detected, otherwise FALSE.
getBarcodeTypeFront()	Int. Represents the barcode type on the front side of the document. If no barcode is detected on the front side, this method returns -1.
getBarcodeDataFront()	String. The parsed data from the barcode located on the front side of the document. If no barcode is detected on the front side, this method returns null.
getBarcodeTypeBack()	Int. Represents the barcode type on the back side of the document. If no barcode is detected on the back side, this method returns -1.
getBarcodeDataBack()	String. The parsed data from the barcode located on the back side of the document. If no barcode is detected on the back side, this method returns null.

**Parse barcode data for Driver's License for US/Canada**

To parse the results from the Driver's License barcode when barcode type PDF\_417\_CODE is detected, use the following method, which will return a dictionary:

```
OcrValidationUtils.DLBarcodeParser.parseDLBarcode (String barcodeData)
```

The dictionary contains 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

### Exception handling

The Library will catch an exception that is thrown by the camera or any other runtime error. The exception details will be sent to the calling app as a string to the `onActivityResult` method. The result type name is `CameraManagerController.RESULT_LIBRARY_ERROR`.

To get the exception details as a string, use the following:

```
String errorMessage =
data.getStringExtra (CaptureIntent.mobiFLOW_ERROR_DETAILS;
```

Then the calling app can decide how to proceed with the error handling and beyond.

The following example code shows how to use the error handling.

```
ActivityResultLauncher<Intent> activityLauncher = registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), new
ActivityResultCallback<ActivityResult>() {
    @Override
    public void onActivityResult(ActivityResult result) {
if (CameraManagerController.sessionType == SessionType.TEST) {
return;
}

switch (result.getResultCode()) {
case RESULT_OK:
// parse session image result.
currentSessionResult = CaptureIntent.parseActivityResult(result.getData());
if (currentSessionResult.getOcrParams() != null) {
String ocrResult = currentSessionResult.getOcrParams()[1];
}
// get the front image size if needed for split capture.
if (currentSessionResult.getFrontImageRectArray() != null) {
inputFrontImageArray = currentSessionResult.getFrontImageRectArray();
}
// use this to get barcode results.
//currentSessionResult.getBarcodeResult();
// save multiple images on MultiCapture mode.
if (!isMultiCapture) {
// the images are received as byte[] on device memory,
// this is helper method to save the images to device file system. if needed
saveImagesToDevice();
}

break;
// user pressed on cancel button
case RESULT_CANCELED:
break;
// user pressed cancel from Alert
case CameraManagerController.RESULT_CANCELED_FROM_ALERT
break;
// get result after session has been closed.
case CameraManagerController.RESULT_CLOSE_SESSION:
// will reach here after
currentSessionResult = captureIntent.parseActivityResult(result.getData());
// decode images, Tiff and jpg
```

```
if (currentSessionResult.getFrontImageRectArray() != null) {
    inputFrontImageArray = currentSessionResult.getFrontImageRectArray();
}
saveImagesToDevice();
break;
// handles camera permissions access denied
case CameraManagerController.RESULT_CAMERA_PERMISSION_ACCESS_DENIED:
// on api 23 target apps, mobiFLOW checking for camera permission
// if not approved will reach here.
String permissionErrorMessage =
data.getStringExtra(CaptureIntent.mobiFLOW_ERROR_DETAILS);
if (permissionErrorMessage != null)
Log.e("error", permissionErrorMessage);
break;

// will get here if error or exception was thrown from the library.
case CameraManagerController.RESULT_LIBRARY_ERROR:
//the exception or error will be received here.
String errorMessage = data.getStringExtra(CaptureIntent.mobiFLOW_ERROR_DETAILS);
if (errorMessage != null)
Log.e("error", errorMessage);
// do something with the error
// got unexpected Error from the Library or exception
break;

// invalid License result code
case CameraManagerController.RESULT_LICENSE_INVALID:
String licenseErrorMessage =
data.getStringExtra(CaptureIntent.mobiFLOW_ERROR_DETAILS);
if (licenseErrorMessage != null)
Log.e("error", licenseErrorMessage);
break;
}
// helper method if needed to remove the images byte array
FileUtils.clearMemory();

// must use this to unregister the listeners
CameraController.unregisterListener();
}
});
```

## Library flow

Most users do not need to change the camera session capture flow section and using of the Library is sufficient.


**i** If you need to change a drawable or string resource, you can add a different resource with an identical name to your project. However, do it with caution, as it will override the one used by the Library.

To change the camera overlay, refer to the `CameraOverlayLayout` class and `mbck_camera_layout.xml`. All the visual overlay elements are available there.

## Clear temporary files

The Library does not save the images on the device by default. To save the images on the device, use the same code as in the function `saveImagesToDevice()`;

When using the `saveImagesToDevice()` function, to delete the images you can use the `FileUtils.clearHighResImages(this)` method. If you did not use `saveImagesToDevice()` but saved the images to a different path, the method `FileUtils.clearHighResImages(this)` will not erase your image files.

 All the above mentioned is implemented in the sample SDK app as an example.

## 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 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 to always set `IsDebug` 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 not recommended to save any images/data available from the SDK to the device of the application build, 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 3

# Set up a custom capture user interface

This chapter explains how to customize the capture user interface.

## Change the look and feel


You can customize the user interface by overriding corresponding resources in the library. Any resource residing in the `res\` folder structure can be overridden by the user application.

Following are a few examples that can be overridden:

- Values (residing in `values\`): string values, styles, dimensions, colors.
- Layouts (residing in `layout\`): Primary layout files that are displayed by the library intents are:

Layout file name	Description
<code>mbck_camera_layout.xml</code>	The layout of the preview screen as shown to the end user. This layout also includes a processing screen named <b>processingOverlay</b> .

- Visual resources (residing in `drawable-... \`): PNG files containing images displayed to the user.

 The components referred to programmatically in the layout XMLs should exist in your customized version of the XML file, otherwise run-time errors are generated.

The procedure is essentially the same for all types of resources.

1. Select files you want to customize from the `res\` folder of the library and paste them into the respective `res\` directories in your project. For value resources, only copy the corresponding lines. The name of the resource should remain the same.
2. Modify the resource.

After compilation, resources from your project will override the respective resources from the library.

Using this method, you can modify all visual aspects of the application. It is not possible to generate additional screen functionality, your modified resource may use only the component callbacks available on the original screen.

There is an alternative view with all elements, where each custom element starts with a custom prefix. See [Additional functionality in Custom view](#) for more information.




## Change icons and captions

You can keep some controls from the preview screen and change the image files and captions of what is shown on the capture screen. To do this, change the following files in the res directory.

File name	Description
logo_watermark.png	The logo of the company
btn_torch.png	The flash icon when not selected
btn_torch_selected.png	The flash icon when selected

You can also change the following icons of the indicators and the frame.

File name	Description
ic_boundary_bottom.png	The bottom boundary of the frame when the check is not found.
ic_boundary_top.png	The top boundary of the frame when the check is not found.
ic_boundary_bottom_v.png	The bottom boundary of the frame when the check is found.
ic_boundary_top_v.png	The top boundary of the frame when the check is found.
ic_boundary_bottom_rl.png	The bottom-left boundary of the frame when the check is not found.
ic_boundary_top_rt.png	The top-right boundary of the frame when the check is not found.
ic_boundary_bottom_v_rl.png	The bottom-left boundary of the frame when the check is found.
ic_boundary_top_v_rt.png	The top-right boundary of the frame when the check is found.
btn_general.9.png	The Cancel button background when not selected.
btn_general_selected.9.png	The Cancel button background when selected.

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

The default messages of the label on top of the camera overlay are TISFlowFrontCaption for front and TISFlowBackCaption for back.

To change these captions at runtime, you must change the messages dynamically in the dynamicStrings hash map keys mentioned above and set new values.

Refer to the following table for the messages to change.

String name	Description
TISFlowPleaseCaptureCheckFront TISFlowPleaseCaptureImage	The label caption at the top of the capture screen when capturing the front side of the check or other document.
TISFlowPleaseCaptureCheck TISFlowPleaseCaptureImageBack	The label caption at the top of the capture screen when capturing the back side of the check or other document. With combined front and back capture, this message is displayed after successful capture of the front.
TISSuccessfulReadingTitle	With combined front and back capture, the title of the message that is displayed after successful capture of the front.
TISFlowPleaseCaptureTheBarcode	The label caption at the top of the capture screen when capturing a barcode.

## Change the text indicators

You can change the text style and background in the `mbck_camera_layout.xml` file.

- `@+id/txtIndicator`: Indicators dynamic and static. TextView when not in capture mode.
- `@+id/txtHoldIndicator`: Capture mode TextView.

In the localization files, 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).
TISFlowIndicatorZoomOut	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 the device towards the barcode.

String name	Description
TISFlowInvalidRotation	Indicator that the phone and document do not have the same orientation.

## Change countdown image view

In `mbck_camera_layout.xml`, change the custom view that inherits from `ImageView`. You can modify `android:id="@+id/counter"`.

You can change the colors and style in the `colors.xml` file.

String name	Description
<code>counter_background</code>	Use to change the circle background color.
<code>camera_counter_color</code>	Use to change the text color.
<code>counter_border_color</code>	Use to change the circle border color. Can get color as a resource (not with <code>#some color</code> ).
<code>countDownStartValue</code>	Use to set the number from which the countdown starts when the counter for taking a still image is displayed. Default value: 2.
<code>countDownStopValue</code>	Use to set the number at which the countdown stops when the counter for taking a still image is displayed. The <code>countDownStopValue</code> must be lower than the <code>countDownStartValue</code> . Default: 0.
<code>counterTextSize</code>	Use to change the counter size.
<code>counterFont</code>	Use to change the counter font with one of the following values: <ul style="list-style-type: none"><li>• BOLD_ITALIC</li><li>• BOLD, ITALIC</li><li>• NORMAL</li></ul>

## DebugRectView

Optionally, you can allow the Library to draw a rectangle over the check on video processing. The rectangle is the frame that the algorithm finds over the check.

Use `app:showCurrentRectangleFound = TRUE` in `com.topimagesystems.ui.DebugRectView` from the XML file.


## Camera overlay color

You can change the color of the outside capture frame using the `camera_overlay_color` value in the `color.xml` file.

Use the following capture intent parameters to change dynamic capture colors.

- validRectPaintStroke
- invalidRectPaintStroke
- validRectPaintFill
- invalidRectPaintFill

String name	Description
validRectFillColor	Fill color for a valid rectangle frame.
validRectStrokeColor	Stroke color for a valid rectangle frame.
invalidRectFillColor	Fill color for an invalid rectangle frame.
invalidRectStrokeColor	Stroke color for an invalid rectangle frame.
grid_line_color	The grid color.

 The validRectFillColor, validRectStrokeColor, invalidRectFillColor, and invalidRectStrokeColor strings are not active. These strings will be removed in a future mobiFlow release.

## Rectangle check frame

When the Library goes into capture mode, you can draw a green rectangle over the check frame by setting the value, drawGreenRectangle="true" in com.topimagesystems.ui.CheckBounderiesView.

## Custom view

The custom view allows you to switch between the mobiFlow classic view and any other UI.

The view contains two XML files: custom\_mbck\_camera\_layout.xml and custom\_mbck\_camera\_manager\_layout.xml.

There are other elements that do not exist in the classic view.

- bannerTop – LinearLayout: Allows you to change the color of the top banner.
- bannerBottom - LinearLayout: Allows you to change the color of the bottom of the banner.

## Guidelines popup

customStaticTxtIndicator: Shown in static text, if further guideline is required. The text will disappear when capture starts.

## Additional functionality in Custom view

You can add UI elements with additional functionality to the custom camera layout (the layout XML file custom\_mbck\_camera\_layout.xml, see [Custom view](#)). An element can be added directly to the XML with a unique ID.

The following code is an example of adding a button:

```
<Button
android:id="@+id/customExtraButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:text="extra button" />
```

For information on adding functionality to your new element, or any other UI elements, see [UI events](#).

## Info screen configuration

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

The popup animates from the top of the screen to the center, in landscape mode.

You can change the text in the integer.xml file.

String name	Description
info_screen_text	The text in the instructions screen

You can also change the animation speed, which is set to 600 milliseconds by default.

String name	Description
customLongAnimation	600


The whole view can be modified in the XML layout-land: info\_screen.xml.

## Start a new activity that is not part of the Library

Declare the activity name to launch in the strings.xml file. The class name should include the full path: package name + class name, as shown in the following example:

```
<string name=" TISCallingAppActivityName">com.topimagesystems.ui.InfoScreenActivity</string>
```

The Library will load the activity from the string.xml entry at runtime. The trigger to open the activity is the onClick method from mbck\_camera\_layout.xml with the method startCallingAppActivity(android:onClick="startCallingAppActivity").

 The activity class must be under the mobiFlow project.

Open a new package name `src` folder and add the activity class to run.

## Leveler configuration

To draw the leveler on every sensor movement, you must disable `android:hardwareAccelerated` in the `cameraController` activity.

Copy the following manifest to your calling application:

```
<activity
android:name="com.topimagesystems.controllers.imageanalyze.CameraController"
android:configChanges="keyboardHidden|orientation|screenSize"
android:hardwareAccelerated="true" >
</activity>
```

## Configure levelerUI

You can change the leveler parameters and location in the `mbck_camera_layout.xml` file.

In the XML file, you can configure the following three custom images:

Image	Description
<code>com.topimagesystems.ui.OneUnitLeveler</code>	One unit leveler inherited from <code>ImageView</code> .
<code>com.topimagesystems.ui.TwoUnitsLeveler</code>	Horizontal image and vertical image inherited from <code>ImageView</code> .
<code>com.topimagesystems.ui.ScaleLeveler</code>	Horizontal image and vertical image inherited from <code>ImageView</code> and contains scale units.

For `TwoUnitsLeveler` and `ScaleLeveler` images, you must specify their location and docking of in the XML file: `left/right` for portrait leveler and `top/bottom` for horizontal leveler.

To disable the leveler, add `android:visibility="gone"`; the leveler will then not be displayed on the screen.

## OneUnitLeveler

The parameters are declared in the `attr.xml` file. You can change the height and the width, as for any Android `ImageView`, with `layout_width` and `layout_height`.

## Leveler parameters

Parameter	Description	Applies to
<code>isFadeOutEnable</code>	True/false fading enable/disable	All leveler types
<code>isDraggingEnable</code>	True/false dragging enable/disable	All leveler types
<code>levelerThickness</code>	Leveler border width (stroke width)	<code>OneUnitLeveler</code>
<code>userColorsInScale</code>	Boolean. Customize scale leveler and set its colors – can be set to multi colors if set to true or single color if set to false.	<code>ScaleLeveler</code>

Parameter	Description	Applies to
scaleUnitGap	The distance between the leveler's units. The number of units is dynamically calculated accordingly.	ScaleLeveler

## TwoUnitsLeveler and Scale Leveler

The leveler height and width are set relative to the capturing frame size (inside red/green boundaries) and determined at run time with the calculation of the screen resolution and the document captured. Therefore, the `layout_width` and `layout_height` will only set the container size, not the image size.

```
levelerLocation: set the leveler location can receive:top,bottom,left,right
levelerLocation="left"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
levelerLocation="top"
    android:layout_width="wrap_content"
    android:layout_height="100dp"
levelerLocation="right"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
levelerLocation="bottom"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

To change the size of the leveler, do the following:

- `paddingLeftAndRight`: For a vertical image, start "x" dp right/left to the frame size (makes the leveler smaller).
- `paddingTopAndBottom`: For a horizontal image start "x" dp from the bottom/top of the screen (makes the leveler smaller).

## Captions and messages

Localization files (`strings.xml`) are available in the resources. You can change the captions and messages used in the Library during the process. By default, English is provided, but you can support different languages in your application if required.

The messages are as follows.

Message Name	Description
TISFlowPleaseCaptureCheckFront	The caption displayed when capturing the front face of the image/check.
TISFlowPleaseCaptureCheckBack	The caption displayed when capturing the back side of the image/check.
TISFlowCancel	The Cancel button is displayed in the error messages.
TISFlowOK	The OK button is displayed in the error messages.
TISFlowPleaseCaptureBarcode	Instruction to capture the bar code in Static capture, when bar code capture is enabled.

Message Name	Description
TISFlowDigitalRowNotInScope (Checks only)	Message when the digital row is not within the set length.
TISFlowCancelCaptureBotton	The Cancel button caption is displayed on the capture screen.
TISFlowPreparingForServerLocating Boundaries	Instruction message displayed for notification OCRNotificationStatusLocatingBounderies.
TISFlowPreparingForServerBinarizing	Instruction message displayed for notification OCRNotificationStatusImageBinarazing.
TISFlowPreparingForServerCropping	Instruction message displayed for notification OCRNotificationStatusImageCropping.
TISFlowErrorReading	Title for all error messages.
TISFlowErrorReadingCheck	Message when mobiFlow failed to read the digital row and the recapture of the front is necessary.
TISFlowErrorReadingImageContrast	Message when there are contrast issues in detecting colors for the digital row reading.
TISFlowErrorReadingImageGeneral TISFlowErrorReadingCheckGeneral	General message about failure to validate the image; displayed if a 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 darker 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 smaller than 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.
TISErrorBlurFail	Message when the image is detected as blurred.
TISFlowWarningMICRDetectedOnCheckBack (Checks only)	Message when the MICR was detected while the user tried to capture the front of the check instead of the back.
TISFlowWarningMicrInterrupted (Checks only)	Message when the recognition of the MICR detects that there is interruption such as stains or the signature is detected in the MICR recognition.



<b>Message Name</b>	<b>Description</b>
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.
TISFlowFrontCaption	Enables reflection on the UI of the front Capture mode.
TISFlowBackCaption	Enables reflection on the UI of the back Capture mode.

## Chapter 4

# 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 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 five images for the front and five for the back are saved, depending on which images you decided to output (see [Handle the session 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 capture, and you are 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. Additionally, you can scan the document and send a copy that the Support Team can print and test themselves.

## Chapter 5

# Guidelines for successful capture

This chapter provides the guidelines that you should follow to ensure successful and optimal capture from the mobiFlow library. These guidelines are not mandatory and a document can still be captured, however, following them will ensure best results.

## Contrast

Position the document on a background with a different color. Use strong visual contrast near the document's boundaries. For documents with multiple colors around the boundaries, use 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. Avoid strong lines on the background that do not belong to the document. Keep the surface around the document clear of any objects about 6" (15 cm) from each side of the document.

## Lighting

Avoid strong direct sunlight or artificial lighting on the document. Avoid having strong light on one part of the document and shade over another part. 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 like that of the document, that is, the picture should be taken in landscape. Position the document at 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, the 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.

## Digital row (MICR): Checks only

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.

## Closing camera

While capturing, it is advisable to close the camera before moving the application to the background for a smoother operation.