



# PhoneGap Plugin

## PhoneGap Plugin Developer's Guide

Version: 3.8.0

Date: 2023-01-23

© 2022 Kofax. All rights reserved.

Kofax is a trademark of Kofax, Inc., 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 Kofax.

# Table of Contents

<b>Preface</b> .....	<b>4</b>
Getting help with Kofax products.....	4
Product documentation.....	5
Default online documentation.....	5
Configure offline documentation.....	5
<b>Chapter 1: Overview</b> .....	<b>6</b>
<b>Chapter 2: Getting started with the PhoneGap Plugin</b> .....	<b>7</b>
Development guidelines.....	7
PhoneGap sample application.....	8
How to use the plugin.....	9
iOS and the PhoneGap Plugin.....	9
Required libraries and frameworks for iOS.....	9
Build the iOS PhoneGap sample application.....	9
Android and the PhoneGap Plugin.....	10
Required updates.....	11
Selecting Camera or CameraX for Android.....	11
Integration of face detection for Android.....	11
Build the Android PhoneGap sample application.....	12
Licensing.....	13
<b>Chapter 3: Using the PhoneGap Plugin with TotalAgility</b> .....	<b>14</b>
KTA connect sample application.....	14
How to use the plugin with TotalAgility.....	14
Kofax TotalAgility Connect sample application.....	15
For iOS.....	15
For Android using Android Studio.....	15
User recommendations for taking a photograph.....	15
Modifying the KTA Connect sample application.....	16
Creating a new PhoneGap Plugin application for TotalAgility.....	16
Licensing.....	16
iOS license key.....	16
Android license key.....	16
Push notification.....	17
Server-Side installation.....	18

# Preface


This guide provides information you need to successfully integrate the PhoneGap Plugin into your mobile project.

## Getting help with Kofax products

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

To access the Kofax Knowledge Base:

1. Go to the [Kofax website](#) home page and select **Support**.
2. When the Support page appears, select **Customer Support > Knowledge Base**.

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

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.  
Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.  
Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.

From the Knowledge Base home page, you can:

- Access the Kofax Community (for all customers).  
Click the **Community** link at the top of the page.
- Access the Kofax Customer Portal (for eligible customers).  
Click the **Support** link at the top of the page. When the Customer & Partner Portals Overview appears, click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).  
Click the **Support** link at the top of the page. When the Customer & Partner Portals Overview appears, click **Log in to the Partner Portal**.
- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.  
Go to the **General Support** section, click **Support Details**, and then select the appropriate tab.

## Product documentation

By default, the Kofax Mobile Capture SDK documentation is available online. However, if necessary, you can also download the documentation to use offline.

### Default online documentation

The product documentation for Kofax Mobile Capture SDK 3.8.0 is available at the following location.

[https://docshield.kofax.com/Portal/Products/en\\_US/KMC/3.8.0-hyeayhcnnoo/SDK.htm](https://docshield.kofax.com/Portal/Products/en_US/KMC/3.8.0-hyeayhcnnoo/SDK.htm)

### Configure offline documentation

To access the documentation offline, download `KofaxMobileCaptureSDKDocumentation-3.8.0_EN.zip` from the [Kofax Fulfillment Site](#) and extract it on a local drive available to your users.

The compressed file includes both `help` and `print` folders. The `print` folder contains all guides, such as the Installation Guide and the Administrator's Guide. The `help` folder contains APIs and other references.

## Chapter 1

# Overview

Kofax Mobile Capture SDK PhoneGap Plugin enables you to add capture, extraction, and processing features to mobile apps. The PhoneGap Plugin uses Apache Cordova, an open-source, multi-platform environment based on HTML, CSS, and JavaScript. This enables the PhoneGap Plugin to support two-way communication with server-based applications. For example, an app can display forms from a TotalAgility server with embedded image capture and bar code controls.

The PhoneGap Plugin exposes most of the Kofax Mobile Capture SDK API within a Cordova application. The plugin code calls existing SDK methods and sends the response back to the `KofaxCordovaPlugin` JavaScript code. `KofaxCordovaPlugin` contains native methods that are called from the app using `kfxMobilePlugin.js`.

The *Kofax Mobile Capture SDK PhoneGap Plugin Developer's Guide* shows you how to create mobile apps with Cordova and applicable Kofax Mobile Capture SDK APIs. It also shows you how to create a sample app that demonstrates the functionality of the SDK. To learn more about Apache Cordova, visit its website at <https://cordova.apache.org/>.


When using Kofax Mobile Capture SDK PhoneGap Plugin, note the following:

- See the *Kofax Mobile Capture SDK Technical Specifications* for supported versions of Cordova.
- Check capture is not supported for mobile check deposit use cases. It may be used for other uses, such as to validate the information on a check to some other document (such as an invoice).
- Kofax Mobile Capture SDK PhoneGap Plugin should not be confused with Adobe PhoneGap. The Kofax Mobile Capture SDK PhoneGap Plugin uses Apache Cordova, not the Adobe product.

## Chapter 2

# Getting started with the PhoneGap Plugin

To get started with the PhoneGap Plugin, extract the files from .zip file. The following files are provided:

File Name	Description
kfxMobilePlugin.js	The plugin APIs are exposed via <code>kfxMobilePlugin.js</code> . All operations are exposed through plugin objects. <div data-bbox="250 768 675 947"><p> There are many additional Javascript files included in the <code>www</code> directory, which contain the classes referred to from the main Javascript file.</p></div>
KofaxMobileSdkPlugin.jar	This is the native part of the Android plugin. This part is responsible for interacting with native libraries.
kfxMobileCordova.framework	This is the native part of the iOS plugin. This part is responsible for interacting with native libraries.
Plugin.xml	This is the main part of the plugin. By using this, Cordova will install the plugin for the iOS and Android platforms.

## Development guidelines

When developing an app with the PhoneGap Plugin, note the following:

- The size of user interface controls may differ across devices.
- Plugin calls are asynchronous. Therefore, you should place actions such as "take picture" and "read bar code" in the success callback of the corresponding `addXXXView` method. Otherwise, on low-end devices, the API may not work as expected.
- All user interface controls are on top of CordovaWebview, including native controls added to an app with any plugin SDK feature.
- Remove unused images to manage memory and prevent app failures. The plugin maintains an image array. Having too many elements (which can be three or more, depending on the device's memory capacity), the app will run out of memory and stop running.
- The output image setting `jpegQuality` specifies the compression quality for the output jpg file created during image processing. This setting is applied only when both of the following are true:
  - The mime type setting (`mimeType`) is set to `MIMETYPE_JPEG`.
  - Representation is set to `IMAGE_REP_FILE`, or `IMAGE_REP_BOTH`.

## PhoneGap sample application

A sample PhoneGap application, demonstrating the plugin, is available in the `\Hybrid\PhoneGap\Samples\Sdk-sample` folder. The sample application can be used on iOS and Android devices.

The sample app demonstrates the capture, processing and extraction features. Mobile ID, Passport, Check Capture, and Bill Pay are four components of the sample app which use these SDK features.

The sample app is modular and has individual JS files (`capture.js`, `processor.js` and `OnDeviceExtraction.js`) for each feature. A user can use any of the JS files and write his/her own application for PhoneGap.

Every component has a corresponding JS file that interacts with the capture, processor and extraction modules. A user can simply reuse any of the components along with these modules and the corresponding application JS file to run it as an independent application.

### **Mobile ID**

If the user selects Mobile ID, the application navigates to the next screen and displays two options: "ID front and back", and "Passport". If the user selects "ID front and back", the application navigates to the country selection screen. After selecting the country, the user is prompted to capture the front and back of the ID.

If the user selects "Back Image", the application shows a popup dialog with three options ("Capture Image", "Barcode", and "Skip"). If the user selects "Capture Image", the application will capture an image and send it for processing. If the user selects "Barcode", the application will capture the bar code, after which the user taps an extraction button to get the extracted data.

The images are sent for extraction, after which the extraction results are displayed.

### **Check Capture**

If the user uses Check Capture, the application navigates to the country selection screen. After selecting the country, the user is prompted to capture the front and back of the check. The captured front and back images are sent for processing as well as extraction, after which the extraction results are displayed.

### **Bill Pay**

If the user selects "Bill Pay", the user is prompted to capture an image of the bill. The application will process the bill and send the captured image for extraction, after which the extracted results will be displayed.

### **Passport**

If the user selects Mobile ID, the application navigates to the next screen and displays two options: "ID front and back", and "Passport". If the user selects "Passport", the user is prompted to capture an image of the passport. The captured image is sent for processing as well as extraction, after which the extraction results are displayed.



### Credit Card

The Credit Card feature can be used to capture both embossed and non-embossed credit cards. The user begins by selecting the credit card component and then captures front of the card. The captured image is then sent for data extraction to the credit card server.

The information is shown to the user, who can then verify and correct as needed before submitting the information. Please note the card is not actually used, since this is a demo application.

## How to use the plugin

You can use the plugin with an existing application or the newly created application. Some changes need to be made to the plugin before adding it to the project. For instructions, go to the `\Hybrid\PhoneGap\Plugins\com.kofax.mobile.plugins.sdk` folder and open `ReadMe.md`.

## iOS and the PhoneGap Plugin

If you are building iOS apps with the PhoneGap Plugin, set up the project as shown in the *Kofax Mobile Capture SDK Developer's Guide* with the additional information shown in this section.

### Required libraries and frameworks for iOS

For an iOS project, the following frameworks are required:

- MobileSDK.framework
- SDKStrings.bundle
- uiimage.bundle

Integrate these libraries into the iOS project by doing the following:

1. Update the application project file to link with MobileSDK.framework.
2. Update the application project file to copy uiimages.bundle and SDKStrings.bundle into bundle resources, so they will be available at run time.
3. If your application uses an older version of SDKAPI, set the Header Search Paths to specify the path to MobileSDK.framework. This step enables the app to locate the `kfxLibEngines`, `kfxLibLogistics`, `kfxLibUIControls`, and `kfxLibUtilities` header files.
4. If there are any existing dylib files, remove them and add the tbd equivalents.
5. Add a dictionary named **App Transport Security Settings** and add a key value pair of **Allow Arbitrary Loads** and **YES** to the info.plist file.

### Build the iOS PhoneGap sample application

For iOS, the sample app is located by default at `/Hybrid/PhoneGap/Samples/SdkSample`. Make the following changes to the sample app before running it.

1. Extract frameworks folders from `/iOS/Frameworks/MobileSDK.zip`.

2. Copy all frameworks folders from `/iOS/Frameworks/MobileSDK` to `SdkSample/plugins/com.kofax.cordova/lib/iOS`.
3. Modify `SdkSample/www/js/app.js` to add the SDK license string. Replace "Add License here" with your mobile SDK license.
4. Open the sample app in Terminal by running the following command:  

```
cd /Hybrid/PhoneGap/Samples/SdkSample
```
5. Add the iOS platform by running the following command in Terminal.  

```
cordova platform add ios
```

An iOS Xcode project is created in the `platforms` folder.

## Run the sample app using Terminal

When running the sample app using Terminal, note the following:

- Make sure that Python has been installed on `/usr/bin/<Python location>` to launch the app on iOS devices.
- Add to development team ID in the build and run commands for iOS as indicated.

Then follow these steps:

1. Build the sample app by using the following command in Terminal:

```
cordova build ios --buildFlag="-UseModernBuildSystem=0" --buildFlag="DEVELOPMENT_TEAM=yourDevelopmentTeamID"
```

Replace `yourDevelopmentTeamID` with the development team ID.

2. Run the sample app by using the following command in Terminal:

```
cordova run ios --buildFlag="-UseModernBuildSystem=0" --buildFlag="DEVELOPMENT_TEAM=yourDevelopmentTeamID"
```

Replace `yourDevelopmentTeamID` with the development team ID.

The app is launched on the device.

## Run the sample iOS app from Xcode

Before running the app, make sure that code signing has been completed. Then do the following:

1. Open the Xcode project in `/Hybrid/PhoneGap/Samples/SdkSample/platforms/ios`.
2. Build and run the project.

## Android and the PhoneGap Plugin

If you are building Android apps with the PhoneGap Plugin, set up the project as shown in the *Kofax Mobile Capture SDK Developer's Guide* with the additional information shown in this section.

## Required updates

If you are developing an Android app, do the following in your PhoneGap Plugin project.

1. Update the `libs` folder in with the latest `.aar` and `.iso` files from the Kofax Mobile Capture SDK. You will find the files in these folders:
  - `\Android\MobileSDK_libs\aar`
  - `\Hybrid\PhoneGap\Plugins\com.kofax.mobile.plugins.sdk\lib\Android`
2. Update the gradle file to include `aar` and `jar` files in the build path.

## Selecting Camera or CameraX for Android

If you are developing an Android app, you can select either Camera or CameraX. See the *Kofax Mobile Capture SDK Developer's Guide* for instructions.

## Integration of face detection for Android

Android Selfie Capture Experience and Quick Extractor have a dependency on the MLKIT in Android. If you want to use one of these features in your application, you need to integrate the MLKIT as a dependency.

There are two ways to integrate MLKIT:

- A bundled model that is part of your app.
- An unbundled model that depends on Google Play Services.

The two models are the same. If you select the unbundled model, the app will be smaller.

### Bundled model

Models are statically linked to your app at build time.

1. Locate `plugin.xml` in `/Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk`.
2. Under the Android platform tag (`<platform name="android">`), replace:

```
<framework src="com.google.android.gms:play-services-vision:10.0.1" />
```

With the following:

```
<framework src="com.google.mlkit:face-detection:16.1.2" />
```

3. Locate `kofax.gradle` at `/Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk`. In the `dependencies` section, replace this line:

```
(implementation fileTree(dir: 'libs/NativeLibs', include: '*.jar'))
```

With the following line:

```
implementation fileTree(dir: 'libs/NativeLibs', exclude: ['javax.inject-1.jar', 'okhttp-3.10.0.jar', 'okio-1.14.0.jar'], include: '*.jar')
```

4. In the application's `config.xml` file, locate the Android platform tag (`<platform name="android">`) and add the following line:

```
<preference name="AndroidXEnabled" value="true" />
```

### Unbundled model

The unbundled model is dynamically downloaded via Google Play Services while installing the application.

1. Locate `plugin.xml` at `/Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk`.
2. Under the Android platform tag (`<platform name="android">`), do the following:

- a. Replace:

```
<framework src="com.google.android.gms:play-services-vision:10.0.1" />
```

With the following:

```
<framework src="com.google.android.gms:play-services-mlkit-face-detection:16.2.0" />
```

- b. Add the following lines:

```
<config-file parent="application" target="AndroidManifest.xml">  
  <meta-data android:name="com.google.mlkit.vision.DEPENDENCIES"  
    android:value="face" />  
</config-file>
```

3. Locate `kofax.gradle` at `/Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk`. In the `dependencies` section, replace this line:

```
(implementation fileTree(dir: 'libs/NativeLibs', include: '*.jar'))
```

With the following line:

```
implementation fileTree(dir: 'libs/NativeLibs', exclude: ['javax.inject-1.jar',  
'okhttp-3.10.0.jar', 'okio-1.14.0.jar'], include: '*.jar')
```

4. In the application's `config.xml` file, locate the Android platform tag (`<platform name="android">`) and add the following line:

```
<preference name="AndroidXEnabled" value="true" />
```

## Build the Android PhoneGap sample application

For Android, the sample app is located by default at `\Hybrid\PhoneGap\Samples\SdkSample`. Make the following changes to the sample app before running it.

1. Go to the `\Android\MobileSDK_libs\aar` folder and copy `arm64-v8a` and `armeabi-v7a` to the `SdkSample\plugins\com.kofax.cordova\lib\Android` folder.
2. Copy all libraries (except `arm64-v8a` and `armeabi-v7a`) from `\Android\MobileSDK_libs\aar` to `SdkSample\plugins\com.kofax.cordova\lib\Android\NativeLibs` folder.
3. Modify `SdkSample\www\js\app.js` to add the SDK license string. Replace "Add License here" with your mobile SDK license.
4. Open the sample app in the Command Prompt by running the following command:

```
cd \Hybrid\PhoneGap\Samples\SdkSample
```

5. Add the Android platform by running the following command in the Command Prompt.

```
cordova platform add android
```

An Android Studio project is created in the `platforms` folder.

6. Do the following to use bar code functionality to `SDKSample.app`.
  - a. Open the following file:

```
SdkSample/platforms/android/com.kofax.cordova/sample-kofax.gradle
```

- b. In the dependencies section, add the following line:

```
implementation 'com.google.mlkit:barcode-scanning:17.0.1'
```

- c. Locate the following line:

```
implementation fileTree(dir: 'libs/NativeLibs', include: '*.jar')
```

- d. Insert the following text in bold to exclude the Kofax library.

```
implementation fileTree(dir: 'libs/NativeLibs', exclude:  
['javax.inject-1.jar'], include: '*.jar')
```

- e. Save the sample-kofax.gradle file.

## Run the sample Android app from the Command Prompt

1. Build the sample app by using the following command in the Command Prompt.

```
cordova build android
```

2. Run the sample by using the following command:

```
cordova run android
```

The app is launched on the device.

## Run the sample Android app from the Android Studio

1. In Android Studio, import the sample project from `\Hybrid\PhoneGap\Samples\SdkSample\platforms\android`.
2. Build and run the project.

## Licensing

When using the plugin in the sample application, you will first need to pass your Mobile SDK license key to the plugin. For the sample application, the license key is set using the `setMobileSDKLicense(callbackSuccess, callbackError, LICENSE_KEY)` method.

In the sample application, this is done in the `app.js` file.

In the sample application directory:

1. Modify `SdkSample\www\js\app.js` to insert your license.
2. Then, either:
  - Use the PhoneGap command line to build your application to ensure the shared javascript is copied to the Android/iOS project folders, or
  - Make the same modifications to the `app.js` files directly in `SdkSample\platforms\android\assets\www\js` and `SdkSample\platforms\ios\www\js`.

Modify this line by replacing `Add License here` with your license key:

```
License.setMobileSDKLicense(function(result)  
{},function(error){alert("error")},'Add License here');
```

## Chapter 3

# Using the PhoneGap Plugin with TotalAgility

Kofax PhoneGap Plugin for SDK makes it possible to access mobile and tablet forms in Kofax TotalAgility, which utilize the new Mobile Capture and Mobile Bar Code Capture controls. By using this plugin in your mobile application, you can use your application to capture and process images and bar code data received from mobile devices.

## KTA connect sample application

A sample PhoneGap application, KTA Connect, demonstrating using the plugin for TotalAgility is available in the `Hybrid\PhoneGap\Samples\KTAConnect` folder. The sample application can be used on iOS and Android devices.

The sample application initially displays a screen that allows the user to enter a remote URL to a Kofax TotalAgility form. After entering the URL, the form is displayed on the screen and functions as if it were loaded in a mobile browser, but with the added capability of using the Kofax Mobile Capture and Kofax Mobile Bar Code Capture controls.

## How to use the plugin with TotalAgility

Follow the instructions in the previous chapter to add the plugin to a new or existing application. You will need to add the mobile SDK frameworks and libraries to the application.

Once the plugin has been added, you can utilize it by navigating a `CDVWebView` (iOS) or `CordovaWebView` (Android) to a remote TotalAgility form.

You will also need to ensure that the remote host is listed in the PhoneGap Plugin white list, and will need to include three query string parameters at the end of the URL, namely `cordovaVersion`, `cordovaPlatform`, and `pushnotifytoken` (e.g. `http://kta.com/TotalAgility/forms/Logon.form?cordovaVersion=8.0&cordovaPlatform=ios&pushnotifytoken=xxxxxxx.`) See discussion below on push notification for what the `pushnotifytoken` should contain, or exclude that parameter if you are not interested in push notifications in your application.

See the sample application for an example on how to navigate to a remote form.

There are a variety of ways to use the plugin in your mobile application depending on your usage.

- Modify the existing sample application.
- Create a new PhoneGap application and add the plugin.
- Add the plugin to an existing compatible PhoneGap application.

- Add PhoneGap and the plugin to an existing mobile application.

## Kofax TotalAgility Connect sample application

See the following instructions for building on iOS, and building on Android.

### For iOS

Copy the SDK framework and bundles to the root directory of the sample app, along with the PhoneGap Plugin Framework (kfxMobileCordova.framework) from: `Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk/lib/iOS/`.

### For Android using Android Studio

Before building, please copy the Cordova plugin files and the SDK libraries manually:

1. Create `/libs` folder in project if not already there.
2. Copy the following file from Hybrid folder location: `Hybrid/PhoneGap/Plugins/com.kofax.mobile.plugins.sdk/lib/Android/KofaxMobileSdkPlugin.jar` into the `/libs` folder.
3. Copy all files from SDK release package located at: `Android/MobileSDK_libs/aar/*` into the `/libs` folder.

## User recommendations for taking a photograph

While using the library to perform camera-based image processing, the results are dependent upon the quality of the original photograph. To ensure that users achieve optimal results, they should be encouraged to follow certain recommendations:

- When possible, set the camera resolution to a minimum of 5 MP or 8 MP for larger documents.
- Do not use zoom. If it is available, it must be set to 1x.
- Flatten wrinkled pages or upturned corners even if they do not include data.
- Place the document on a flat non-cluttered surface. This surface should have a distinct, relatively uniform background with as little variation as possible. Avoid backgrounds that look too much like the border areas of the document itself. Desk surface texture is OK, but sharp colors or brightness differences in the background cause problems for page detection.
- Avoid shadows.
- Check the lighting before taking a photo. Good uniform illumination will help to get a faster shot without motion blur and avoid jitter noise because of insufficient light.
- Avoid using flash which can over saturate the picture or wash out a part of the image.
- Maximize the area within the image frame occupied by the document, but make sure that there is a small margin of background surrounding the document. For a standard letter-size page this margin should be about 0.5", for documents of other sizes it should be proportionately smaller or larger.

- Rectangular overhead camera shots are best, but in order to avoid shadows cast by the camera itself it is OK to take the picture from an angle - resulting keystone distortions will be corrected. However, larger angles should be avoided - not because of larger keystone distortions (these can be corrected for most angles), but because of limited depth of field. The rule of thumb is that the depth of field is 27 mm (just over 1 inch) for a picture taken from a distance of 1 foot. So, if the difference between the distances to the most distant point and the closest point exceeds the depth of field, some parts of the document will be blurred.
- If available, use the touch focus feature to focus on the center of the document (or the center of the area of interest).

## Modifying the KTA Connect sample application

Simply load the sample application in the appropriate developer tool and make any desired changes to change branding or add additional UI screens and functionality.

If building for Android, you need to have the Google Play Services SDK installed. Also, due to some deeply-nested image files included by Google Play Services, you should locate the source of your project near the root folder of the drive you are building from, or you may encounter some build error containing the text "No resource found that matches the given name."

## Creating a new PhoneGap Plugin application for TotalAgility

See [Getting started with the PhoneGap Plugin](#) for instructions on how to create a new PhoneGap Plugin application and add a plugin to an application. After generating the application, add the `com.kofax.mobile.plugin.sdk` plugin by referencing the `plugin.xml` file located in `Hybrid \PhoneGap \Plugins\com.kofax.mobile.plugin.sdk\plugin.xml`.

## Licensing

When using the plugin in an application, you will first need to pass your mobile SDK license key to the plugin.

### iOS license key

For iOS, the license key is set using the `[KFXMobileSdkPlugin setLicenseKey: key]` message. In the sample application, this is done in the `RemoteUrlViewController.m` file:

```
[KFXMobileSdkPlugin setLicenseKey:@"MYLICENSE"];
```

Modify this line, replacing MYLICENSE with your license key.

### Android license key

For Android, the license key is set using the `MobileSdkPlugin.setLicenseKey(String key)` method. In the sample application, this is done in the `TotalAgility.java` file:



```
// Set the license key  
MobileSdkPlugin.setLicenseKey(LICENSE_KEY);
```

Modify this line:

```
public static String LICENSE_KEY = "MYLICENSE";
```

Replacing MYLICENSE with your license key.

## Push notification

The KTACoconnect TotalAgility Sample app also contains sample code to demonstrate how push notifications might be integrated into the workflow of a hybrid/TotalAgility based solution. If you wish to use push notifications, follow the platform-specific procedures to enable push notification support for your application. In iOS, this is called Apple Push Notification (APN) service, and the process involves creating specific certificates and provisioning profiles for your application. For Android, it is "GCM" (Google Cloud Messaging), and involves enabling support and creating an API key and configuration file in your Google developer console.

Once you have these created, the process of adding support to the sample app is as follows:

**iOS:** Be sure to change the app ID to the specific one you created in your Apple developer portal that has been enabled for Push notifications. Build with the corresponding provisioning profile and deploy the matching certificate and private key to your Web server that will be sending the notifications.

**Android:**

1. When you create the API key and configuration for your app in the Google developer console, note the API key, the sender ID, and download the google-services.json configuration file.
2. Copy the google-services.json file into the root directory of the project (where the Android.manifest file is). Modify `RegistrationIntentService.java` (`KTACoconnect\android\src\com\kofax\mobile\sdk\phonegap\TotalAgility`) to replace the `GCM_SenderID` value with your sender ID.
3. Finally, use the API key on the server side when it sends messages through GCM.

Note that the TotalAgility sample will pass the push notification token, if available from the device, to the server as part of the URL. So any TotalAgility form that is navigated to within the embedded Web browser of this sample app will pass a query string variable called "pushnotifypoken" with the device's token, which the TotalAgility code can then use to send messages to the device in the future.

In this sample app, when the notification comes in, if the app is in the foreground, an alert will appear with the title and message body that the server sent along with a redirect URL. If the user presses "cancel", the alert will simply close. If they press "view", the Web view will redirect to the new URL.

If the app was in the background when the notification comes in, it will appear as a system notification - the banner message in iOS, or a vibrate on Android. If you pull down on the

notification area in either platform, the message appears, and touching it will launch the TotalAgility sample app and navigate to the new URL.


## Server-Side installation

On the server side we have provided a helper DLL (`Kofax.Mobile.Notification.dll`) that makes it easier for your Kofax TotalAgility workflow to send notifications back to the mobile device. This DLL, as well as sample forms, can be found in the Mobile SDK: `Hybrid/Phonegap/Samples/KTACoconnect/server` and must be installed as follows:

### Configuring the Server

1. Install the DLL in TotalAgility.
  - Log in to the TotalAgility designer.
  - Navigate to the Integration section.
  - Select .NET Assemblies.
  - Upload the DLL and place it in the KTA .NET AssemblyStore.
2. For iOS, install Certificates by importing the APN (Apple Push Notification) certificate into the Windows certificate store.

### Configure Your Form to Send Notifications

 Sample forms can be found in `Hybrid/Phonegap/Samples/KTACoconnect/server`.

1. Create a .NET Method action on your form.
2. Reference the assembly uploaded in the previous section.
3. Select the `KTAPushNotification.NotificationSender` class.
4. Select the `SendNotification` method.
5. Configure the parameters as needed.
  - `platform`: Set to iOS or Android.
  - `title`: Title of the notification.
  - `message`: Message body of the notification.
  - `pushnotifytoken`: Notification token from your app required to send notifications.
  - `androidAPIKey`: Android application API key.
  - `iOSCertificateName`- is the name of your Apple certificate, e.g., "Apple Production iOS Push Services: com.example.Application."
  - `isAPNProduction` - indicates whether or not you are using a production or development certificate for Apple.
  - `redirectURL` - is the URL to include in the notification body.

Return `Parameter (bool)`; `true` if notification sent successfully or `false` if there is an error with the server sending the notification, such as the inability to reach the device, the user has disabled notifications, or uninstalled the application.

**i** If you are using the Sample Push Notification Forms [NotificationTest.form and NotificationConfirmation.form] you need to import the forms into KTA and then fill in the values for the following Global Variables according to the parameter configuration notes above:

- [PUSH\_NOTIFICATION\_ANDROIDAPIKEY]
- [PUSH\_NOTIFICATION\_IOS\_CERTIFICATENAME]
- [PUSH\_NOTIFICATION\_IS\_APNPRODUCTION]
- [PUSH\_NOTIFICATION\_REDIRECTURL]

(update the value with name of your KTA server)

### Configure a process to send notifications

1. Create a .NET Activity in your process map.
2. Configure it to use the assembly uploaded in the previous section.
3. Configure the parameters (same parameters as above).

### Additional recommendations

When loading forms in the KTA Connect application, the initial form loaded by the app contains multiple query parameters: `cordovaPlatform`, `cordovaVersion`, and `pushnotifytoken`. Note that these parameters are case sensitive.

It is recommended that you create form initialization variables with these names for any mobile/tablet form that will be loaded in your mobile application. These should be passed through any form redirects to ensure that the application passes them through properly and that they are accessible in any form actions. When sending a notification in a process map, this data also needs to be available to the job.