Kofax PhoneGap Plugin

Developer's Guide

Version: 3.3.0

Date: 2018-02-23

Table of Contents

Preface

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

Getting help for Kofax products

Kofax regularly updates the Kofax Support site with the latest information about Kofax products.

To access some resources, you must have a valid Support Agreement with an authorized Kofax Reseller/Partner or with Kofax directly.

Use the tools that Kofax provides for researching and identifying issues. For example, use the Kofax Support site to search for answers about messages, keywords, and product issues. To access the Kofax Support page, go to www.kofax.com/support.

The Kofax Support page provides:

- Product information and release news

  Click a product family, select a product, and select a version number.
- Downloadable product documentation

  Click a product family, select a product, and click **Documentation**.
- Access to product knowledge bases

  Click **Knowledge Base**.
- Access to the Kofax Customer Portal (for eligible customers)

  Click **Account Management** and log in.

  To optimize your use of the portal, go to the Kofax Customer Portal login page and click the link to open the *Guide to the Kofax Support Portal*. This guide describes how to access the support site, what to do before contacting the support team, how to open a new case or view an open case, and what information to collect before opening a case.
- Access to support tools

  Click **Tools** and select the tool to use.
- Information about the support commitment for Kofax products

  Click **Support Details** and select **Kofax Support Commitment**.

Use these tools to find answers to questions that you have, to learn about new functionality, and to research possible solutions to current issues.

Related documentation

Documentation for this product is available online.

Chapter 1

Overview

The Kofax PhoneGap Plugin *Developer's Guide* provides an overview of developing Cordova applications using the Kofax Mobile Plugin for SDK. This document includes information on creating a sample app that can be used to demonstrate the functionality of the SDK. By adding the plugin to your mobile app, it can be used to capture, process and extract documents and bar code data received from mobile devices.

Additionally, there is information for using the plugin in a Cordova application to display forms that come from a TotalAgility server, in order to display embedded image capture and bar code controls.

Note The earliest Cordova version supported by the Kofax PhoneGap Plugin is version 5.0. The latest Cordova version supported by the Kofax PhoneGap Plugin is version 7.0.1.

Chapter 2

The Kofax PhoneGap Plugin for SDK

PhoneGap is an open source mobile application development framework, based upon the Apache Cordova project. See `cordova.apache.org` for documentation for details. The Kofax PhoneGap Plugin for the mobile SDK in your mobile application can be used to capture and process images and bar code data received from mobile devices.

Note PhoneGap is the Adobe branded version. Cordova is the generic open source Apache version. For the purposes of this guide, all references to "PhoneGap" can apply to either PhoneGap, or Cordova.

The plugin exposes much of the Kofax Mobile SDK functionality, 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 a hybrid application via `kfxMobilePlugin.js`

The following PhoneGap related files are provided:

File Name	Description
`kfxMobilePlugin.js` **Note** There are many additional Javascript files included in the `www directory`, which contain the classes referred to from the main Javascript file.	The plugin APIs are exposed via `kfxMobilePlugin.js`. All operations are exposed via the plugin objects:
`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.

The following should be taken into account when building an application using the plugin:

- Size of the UI control displayed may not be the same across all devices.
- Plugin calls are asynchronous. Consequently, it is good practice to put actions like "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 UI Controls will float on top of CordovaWebview. The same thing is true for any native controls added to an application using any plugin SDK feature.

- The developer has to manage memory issues. The plugin maintains an image array. If the image array has more than about 3 elements (depending on device memory capacity) the application may crash. Developers must be sure to remove unused images.
- 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 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\Sdksample` 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 Deposit 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 Deposit

If the user selects "CheckDeposit", 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.

Building the Android PhoneGap sample application

For Android, to build the application first import the project file into Android Studio. Be sure to import the project file as "Existing Android code into workspace". If you import it as a general project, it may not build.

By default the Android project will be located at `\Hybrid\PhoneGap\Samples\Sdksample\platforms\android`.

Note The `www/js/app.js` file needs to be modified with an SDK license string. Replace "Add License here" with your mobile SDK license.

Adding a localized SDK in Android Studio

Please refer to *Getting Started with the SDK* chapter of the *Mobile SDK Developers* guide, for specific library and frameworks to include in order to use the Mobile SDK with your application.

Run the sample app in Android Studio

1. Open Android studio and import the sample project. Select `\Hybrid\PhoneGap\Samples\Sdksample\platforms\android`.
2. Android Studio will display an alert saying either the project doesn't have gradle configurations, or the gradle version is different and allow for automatic generation or update. In either case, click OK. Then, manually edit `build.gradle` for the correct gradle version. (The latest is preferable. Optionally, if gradle and Android studio has been upgraded to latest versions (Android Studio 3.0.0 or higher), then follow the instructions from the Android developer sites `https://developer.android.com/studio/releases/gradle-plugin.html` and `https://developer.android.com/studio/build/gradle-plugin-3-0-0-migration.html` to fix any build errors.). Also, sometimes the `minSDK` version or other configurations needs to be changed in the manifest file.
3. Copy libraries from `Android\MobileSDK_libs\aar` to `libs` in the Android module.
4. Open the `build.gradle` in the Android module and add the following lines in dependencies { .... }:

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

```
}
compile (name: 'sdk-release', ext: 'aar')
```

5. Open the build.gradle module and add the following lines in `Android { .... }`

```
packagingOptions {
 exclude 'META-INF/LICENSE.txt'
 exclude 'META-INF/NOTICE.txt'
 exclude 'META-INF/DEPENDENCIES'
 exclude 'META-INF/LICENSE'
 exclude 'META-INF/NOTICE'
}
```

6. Do a gradle sync.
7. Build and run.

Building the iOS PhoneGap sample application

1. Open the iOS sample application located at `/Hybrid/PhoneGap/Samples/SdkSamples/platforms/ios/` using XCode.
2. By default kfxMobileCordova.framework is added to the application. Make sure that Framework Search Paths in Build Settings of XCode contains `Kofax PG/Plugins/com.kofax.cordova`.
3. Add Kofax frameworks (MobileSDK.framework, uiimages.bundle, SDKStrings.bundle) to the application. Make sure that Header Search Paths in Builds Settings of XCode contains `MobileSDK.framework/Headers`.
4. Build the application by signing with your certificate.

Note The `www/js/app.js` file needs to be modified with an SDK license string. Replace "Add License here" with your mobile SDK license.

Note To avoid a linker error, be sure to include the following in the build instructions for your own application: `-force_load $(BUILT_PRODUCTS_DIR)/libCordova.a`

How to use the plugin

Follow one of the sections below to add the PhoneGap plugin to a new or existing app. You will need to add the Mobile SDK frameworks and libraries to the application. 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.

Modifying the existing sample application

Download the sample app from this folder: `Hybrid\PhoneGap\Samples\Mobile\SdkSample` (or from the website, depending on your particular SDK distribution).

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

Creating a new PhoneGap application

First verify you are using a version of PhoneGap that is compatible with the Mobile Plugin. Read the PhoneGap documentation for instructions on how to create a new PhoneGap 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`.

Adding the plugin to an existing compatible PhoneGap application

First verify the PhoneGap application uses a compatible version of PhoneGap. If it does, follow the PhoneGap documentation instructions to 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`.

Required library and framework files

Please refer to *Getting Started with the SDK* chapter of the *Mobile SDK Developers* guide, for specific library and frameworks to include in order to use the Mobile SDK with your application.

Requirements

Certain frameworks must be included in your project.

For iOS

For an iOS project, the following frameworks are required.

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

In order to integrate the Mobile SDK into an iOS app, you must:

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. (Required only if your application uses an older version of SDKAPI.) In order to successfully locate the `kfxLibEngines`, `kfxLibLogistics`, `kfxLibUIControls`, and `kfxLibUtilities` header files, the application project file needs to be update to specify the location of `MobileSDK.framework` in the Header SearchPaths section.

4. Required for Xcode 7 and above.
   - If there are any existing dylib files, remove them and add the tbd equivalents.
   - Add a dictionary named 'App Transport Security Settings' and add a key value pair of 'Allow Arbitrary Loads' & 'YES' to the `info.plist` file.

For Android

If you want to apply this improvement for your PhoneGap app please update following files:

Update the libs folder in the Phonegap App with latest MobileSDK aar libs, including the *.so files. You can find these libs under `...\Android\MobileSDK_libs\aar and ...\Hybrid\PhoneGap\Plugins\com.kofax.mobile.plugins.sdk\lib\Android`.

Also update gradle file to include aar and jar files into build path.

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 Kofax mobile 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.

The TotalAgility sample application is based on Cordova 5.4. Versions of TotalAgility server prior to version 7.3 does not yet include the Javascript files for Cordova 5.4. In order to patch the server, you need to copy the contents from the following location in our MobileSDK: `Hybrid/Phonegap/Samples/KTAConnect/server/ktaserver_cordova54patch/5.4` to `C:\Program Files\Kofax\TotalAgility\Agility.Server.Web\Forms\Plugins\MobileCapture\Cordova\5.4)`. Then add an initialization variable to your forms that will pass the version number used in the client through.

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 KTA form.

You will also need to ensure that the remote host is listed in the PhoneGap 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=5.4&cordovaPlatform=ios&pushnotifytoken=xxxxxx`.) 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.

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 application for TotalAgility

Read the PhoneGap documentation for instructions on how to create a new PhoneGap 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 KTAConnect 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 (`KTAConnect\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 "pushnotifytoken" 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/KTAConnect/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

Note Sample forms can be found in `Hybrid/Phonegap/Samples/KTAConnect/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` - the title of the notification.
   - `message` - the message body of the notification.
   - `pushnotifytoken` - notification token from your app required to send notifications.
   - `androidAPIKey` - is your Android applications 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.

Note 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.