Mobile SDK

HTML5 SDK Developer's Guide

Version: 3.4.0

Date: 2018-10-11

KOFAX

Table of Contents

Preface

This guide includes the information you need to successfully integrate HTML5 SDK components into your mobile project.

For additional details on API library properties and settings, refer to the HTML5 SDK API Reference Guide.

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.

Product documentation

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

Default online documentation

The product documentation for Mobile SDK 3.4.0 is available at the following location.

https://docshield.kofax.com/Portal/Products/en_US/KMC/3.4.0-o6num87075/SDK.htm

Configure offline documentation

To access the documentation offline, download KofaxMobileSDKDocumentation-3.4.0_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

About the HTML5 SDK

Introduction

This document is intended to provide a brief overview of the usage and features of the HTML5 SDK.

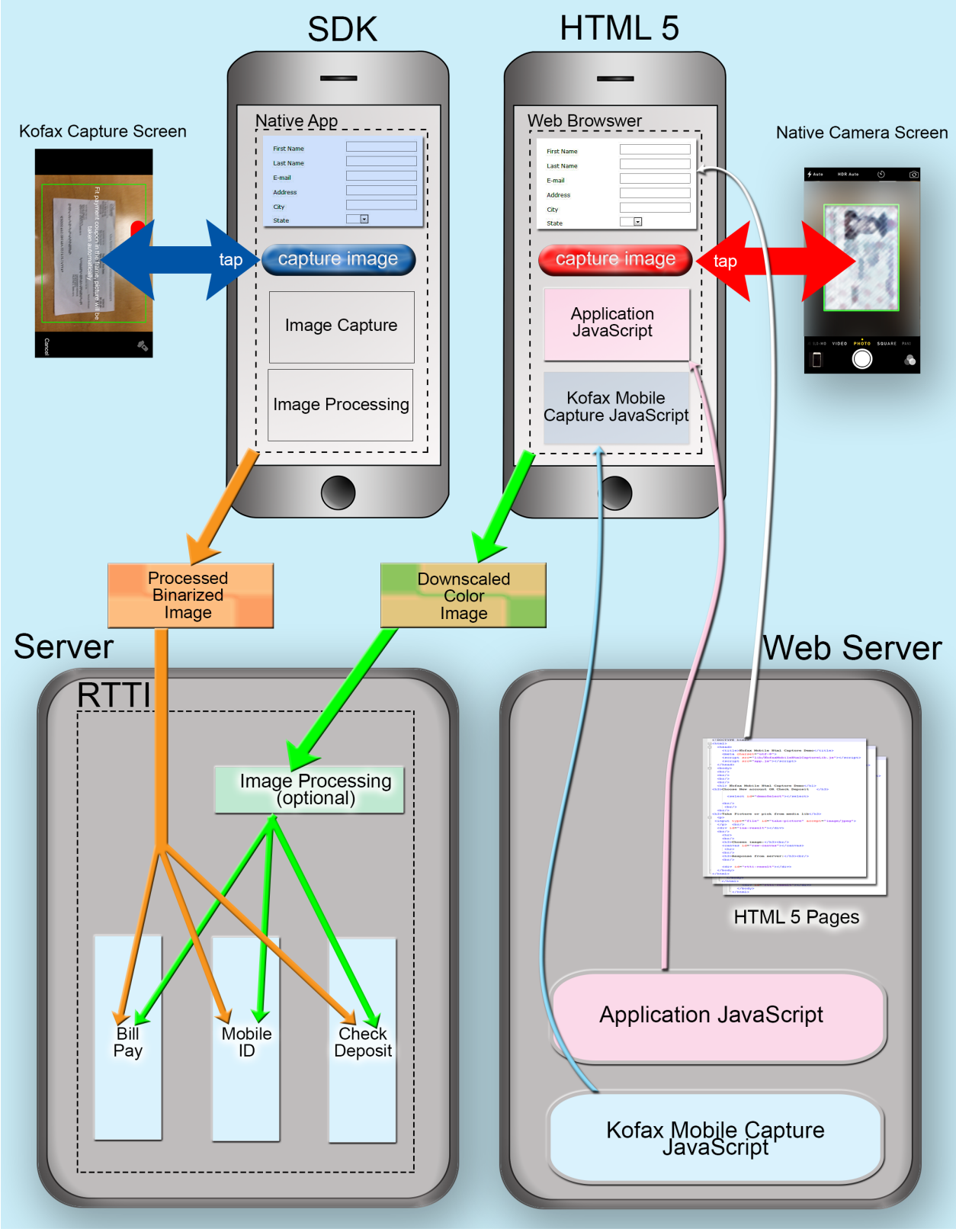
HTML5 Capture

The mobile capture SDK is implemented as a native code library for iOS and Android. As such, it requires the end user to install an application on their device before they can use any of the functionality it offers. However there are some use cases where you may wish to leverage some of the functionality offered in the Kofax Mobile Capture platform (the Kofax Mobile Capture platform includes both the Mobile SDK and the mobile capture framework applications such as: Kofax Mobile Deposit Capture, Kofax Mobile Bill Pay, and Kofax Mobile ID Capture) without requiring the user to download and install an application on their mobile device. To support these use cases, version 2.3 and later of the Mobile SDK includes tools that make it easy for you to build thin-client, HTML5 applications. This allows you to create mobile applications that leverage the mobile capture frameworks without requiring your users to install an application on their device.

One example of the type of use case that lends itself to an HTML5 implementation is opening a new account. In this use case a company wishes to sign-up as many new customers as possible for some sort of account. These prospective customers are directed to an HTML page, perhaps via an advertisement that they receive in their email. If these prospective customers are required to download a new application before they can sign up, it is unlikely that they will follow through and open a new account. In the ideal case the user would be presented with a simple HTML5 user interface that runs in the Web browser on their mobile device. They could use this user interface to sign up. Once they are signed up as a customer, various inducements can be used to get the new customer to install an application which would provide them with a richer and more easy to use interface.

In order to satisfy use cases like the new account opening use case described above, the mobile SDK is offering a new HTML5 capture functionality starting with the 2.3 release. Our implementation consists of a JavaScript library that contains a class called `KfxMobileCapture` that simplifies the process of compressing an image and preparing it to be sent via an HTTP request. Because our native code image processing library cannot be used from an HTML5 application, we have also implemented all of our image processing within Real-Time Transformation Interface and Kofax TotalAgility.

The basic architecture of Kofax's HTML5 solution is shown in the following figure.


SDK
HTML 5
Kofax Capture Screen
Native App
First Name
Last Name
E-mail
Address
City
State
tap
capture image
Image Capture
Image Processing
Web Browswer
First Name
Last Name
E-mail
Address
City
State
capture image
tap
Application JavaScript
Kofax Mobile Capture JavaScript
Native Camera Screen
Processed Binarized Image
Downscaled Color Image
Server
Web Server
RTTI
Image Processing (optional)
Bill Pay
Mobile ID
Check Deposit
HTML 5 Pages
Application JavaScript
Kofax Mobile Capture JavaScript

Compare an HTML5 architecture to a native application. Notice that in the native application, the Kofax capture experience (with auto-capture) is used on the client and all image processing is also done on the client. The processed image is then sent to the server. In the case of the HTML5 application, the camera application for the platform is used to capture an image. The image is downscaled (which reduces the size of the image) using Kofax-provided JavaScript and then is sent to the server in an HTTP request with parameters that indicate that image processing should be done on the Real-Time Transformation Interface and Kofax TotalAgility. On the Real-Time Transformation Interface server, all specified image processing is done first and then the image is sent to one of the Kofax Mobile Capture Frameworks. In reality the HTML5 Web pages and the JavaScript can be hosted on the same server as Real-Time Transformation Interface and Kofax TotalAgility, and the Kofax Mobile Capture frameworks or they can be hosted on different servers as shown in the figure.

To create an HTML5 application you will create a mobile-friendly HTML5 Web page that contains the functionality of your application. This page is hosted on a Web server along with application-specific JavaScript that you write to implement your application (most non-trivial HTML5 applications contain some JavaScript). The JavaScript library should also be deployed onto the Web server and the HTML5 page will need to reference both JavaScript files.

In a typical scenario the user will receive a promotional email. The email will contain a link that will direct them to the Web page described above. When they click on the link the page will be downloaded to their device and rendered within the Web browser installed on their mobile device. Any JavaScript files that the HTML5 page references will also be downloaded to their device. Because this JavaScript is the only code that will execute on the device, the native code and the image capture experience (which includes the ability to do auto-capture) will not be available. Instead you will use the Media Capture API which is part of the HTML5 standard. The HTML5 Media Capture API adds many new syntactic features to HTML5 including the ability to specify that the browser should capture media of a specific MIME type using the media capture capabilities of the hosting device. For example, the following statement in an HTML5 page will use the devices native capabilities for capturing a jpeg image and assign it to a variable called "take-picture" which can be accessed from JavaScript code.

```
<input type="file" id="take-picture" accept="image/jpeg">
```

On most devices, the media capture capability will display the device's native camera application. You can find out more about the HTML5 Media Capture API at the w3.org website.

After the image is captured the user can write JavaScript logic to compress the image, to reduce the bandwidth required to transmit it to the server and then to convert it to base64 format so it can be easily transmitted to the server as part of an HTTP request. Strictly speaking, down-scaling is used, not compression. The JavaScript library includes methods that make this simple. The JavaScript library also includes methods that make it simple to submit a request to one of the Kofax mobile frameworks: Remote Check Deposit, Bill Pay, and Mobile ID. Advanced processing operations must be done on the server as shown in the figure.

An important, final point is that HTML5 is not limited to the iOS and Android platforms. Any platform/browser combination that supports the HTML5 Media Capture API can now be used as a Kofax Mobile Capture platform client.

HTML5 Selfie Capture

The Selfie Capture Experience displays messages to guide the user to take a intelligible Selfie. It is designed to perform a liveness check by looking for eye blinks.

During use, the user is guided by a series of text messages that appear on the screen.

- `CenterFaceMessage`: Appears when the user face is not centered.
- `MoveCloserMessage`: Appears when the MinimumFaceSize Criteria not met.
- `BlinkMessage`: Appears after the face is properly aligned in the target frame and instructs the user to blink.
- `HoldSteadyMessage`: Appears when all criteria is met.
- `DoneMessage`: Appears after a selfie is successfully captured.
- `outOfFrameColor`: Sets the color of the target frame outer view.
- `frameColor`: a property that sets the color of the target frame border.
  The Selfie Capture Experience has configurable selfie detection properties, which include `MinimumFaceSize`, `CenterToleranceFraction`, `FrameAspectRatio`, and `FramePadding`.

HTML5 setup

You need the following items to make use of the HTML5 Capture capabilities of the SDK:

- Web Server (e.g., IIS)
- Text editor

Capture images

When using HTML5 to capture images, image processing should be enabled on Real-Time Transformation Interface and Kofax TotalAgility for optimal results. The administrator's guides for Real-Time Transformation Interface and Kofax TotalAgility describes how to configure image processing on a per-project basis. The various smart mobile components include recommended image processing strings for server-side processing.

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).

HTML5 extraction setup

You need the following items to make use of the HTML5 Extraction capabilities of the SDK:

- Real-Time Transformation Interface server or Kofax TotalAgility server and its associated prerequisites.
- Desired Smart Mobile Components (SMCs). For example, Kofax Mobile Deposit Capture, or Kofax Mobile ID Capture.

WeChat requirements

If you use WeChat, note the following requirements for devices:

- The following OS versions are required:
  - Android: Version 5.0 and above
  - iOS: Version 10.0 and above
- WeChat does not support Advance Capture and Selfie Capture Experience.
- Some devices share camera instances with the front and back cameras. This can cause the back camera to open in the Onboarding app even if the native camera was set to the front-facing camera.
- If the camera does not open, you may need to set permissions for the camera manually in WeChat.

For more requirements, refer to the product Technical Specifications.

Setting up the prerequisites

There are certain configuration steps that must be performed on your server before you can use the HTML5 Capture feature, as explained in the following steps:

1. Ensure that the Real-Time Transformation Interface server is installed with the desired Smart Mobile Components configured and functioning.
2. Configure the Real-Time Transformation Interface server to allow cross-origin resource sharing (see http://en.wikipedia.org/wiki/Cross-origin_resource_sharing) for your HTML5 application
   a. Open Internet Information Services (IIS) Manager.
   b. Select the Real-Time Transformation Interface server application under the specified Web site (e.g., `Default Web Site\mobilesdk`).
   c. Open the Configuration Manager.
   d. Select `system.webServer/httpProtocol` under "Section."
   e. Select "`customHeaders`" and click on the " **...** " button.
   f. On the right side, click Add and enter the name/value for these three pairs
      - `Name: Access-Control-Allow-Origin; Value: *`
      - `Name: Access-Control-Allow-Headers; Value: Content-Type`
      - `Name: Access-Control-Allow-Methods; Value: PUT, POST, GET, OPTIONS`

`Access-Control-Allow-Origin`	`*`
`Access-Control-Allow-Headers`	`Content-type`
`Access-Control-Allow-Methods`	`PUT, POST, GET, OPTIONS`

   g. Close the editor and select **Apply** under **Actions**.

Note Configure Kofax TotalAgility server by follow the above steps.

Using the HTML5 SDK with other HTML5 applications

To create a new HTML5 application and use or integrate HTML5 SDK, the app developer needs to follow the below instructions.

1. Create an HTML5 application.
2. Include the SDK .css file in the application HTML files.

   Add the following code there: `<link rel="stylesheet" href="../../KfxWebSDK/CSS/KfxWebSDK.css">`. Be sure to change the path to `KfxWebSDK.css` according to your configuration (SDK location).
3. Include SDK java script minified file. Add the following code there: `<script src="../../KfxWebSDK/KfxWebSDK.js"></script>`. Be sure to change the path to `KfxWebSDK.js`

according to your configuration (SDK location). This file contains all necessary 3rd party libraries, so there is no need to worry about any SDK dependencies.

> **Note** Do not move or rename anything in the SDK folder.
> There are several directories in the SDK main folder (`KfxWebSDK`) such as the CSS, Resources, Images, and so on. Do not change the directory structure of the HTML5 SDK and do not rename the files. Doing so may break the SDK.

4. To ensure the SDK content is loaded successfully, or to debug any issues, please use the Web Developer Tools and console. You can find this view in most popular browsers. You can also debug remotely on a device. Please refer to the browser's user guide. For example, here is the link to a description of the Chrome remote debugging process: `https://developer.chrome.com/devtools/docs/remote-debugging`.

HTML5 SDK external classes

HTML5 SDK has the following external classes:

- KfxWebSDK.Capture
- KfxWebSDK.SelfieCapture
- KfxWebSDK.DocumentExtractor
- KfxWebSDK.ReviewControl
- KfxWebSDK.ImageProcessor
- KfxWebSDK.Utilities
- KfxWebSDK.AppStats (Kofax AppStats)

The following sections describe these classes in detail.

KfxWebSDK.Capture (Kofax Capture)

This class provides methods to capture a document either from a camera or photo library. It enhances the user experience by adding feedback while the user captures a document. This guidance makes it easier to capture high quality images.

Native

Package name: `com.kofax.capture`

Global Namespace: KfxWebSDK

Class Name: Capture

JavaScript Closure

KfxWebSDK.Capture

APIs

API	Parameters	Description
create	options successCallBack errorCallBack	Creates a Capture control based on given options. It will always use the rear camera. `Options.containerId`: Empty divId, where the application developer wants to see a camera preview along with capture guidance. The div container must exist and be empty, otherwise an error will be thrown. The application developer has to properly set the size and position of the div. The SDK doesn't check the size and position or any other container css properties, this is a developer responsibility. `Options.preference`: camera/gallery, from where the developer would like to capture a document. `Options.preview`: Boolean value representing whether or not to review the captured image using the SDK review control. In case of FALSE, the developer needs to implement its own review functionality. This option effects only web capture, when the captured image is from the gallery via the native camera there is no review screen available. `Options.videoStream`: Boolean representing to follow either the standard capture or document capture process. Various capture criteria options can be set here as well (see setOptions below). If you do not set any capture criteria options here, the default values will be used (see getDefaultOptions below). **Note** The requirement to choose the gallery is a limitation in both Android and iPhone. Camera only is a limitation in iPhone.

API	Parameters	Description
setOptions	options successCallBack errorCallBack	Sets various capture criteria. `{ useTargetFrameCrop: false, frameAspectRatio: 0.628, framePadding: 5, frameCornerHeight: 15, frameCornerWidth: 70, frameCornerColor: '#00FF00', outOfFrameTransparency: 0.5, showEdges: false, edgesColor: '#FFFF00', edgesWidth: 4, guidanceSize: 150, criteria: { captureTimeout: 1700 centerToleranceFraction: 0.15 longAxisThreshold: 85, shortAxisThreshold: 60, maxFillFraction: 1.8 minFillFraction: 0.65 turnSkewAngleTolerance: 10, pitchThreshold: 15, rollThreshold: 15 }, lookAndFeel: { documentSample: 'http://example.com /images/document_sample.jpg', forceCapture: 10, gallery: true } }`
getOptions	successCallback errorCallback	Returns current capture control options for capture criteria, capture guidance messages and other configurable ui options. `successCallBack`: callback with JSON object representing capture control options. `errorCallBack`: callback with error message to be invoked when something goes wrong.
getDefault Options	successCallBack errorCallBack	Returns default capture control options for capture criteria, capture guidance messages, and other configurable UI options. `successCallBack`: callback with JSON object representing capture control options. `errorCallBack`: callback with error message to be invoked when something goes wrong.
takePicture	successCallBack errorCallBack	Starts the Auto Capture process `successCallBack`: callback with ImageData representation of the captured image. `errorCallBack`: callback with the error message to be invoked when something goes wrong.

API	Parameters	Description
takePicture Continually	successCallBack errorCallBack	Starts Continuous Auto Capture process. `successCallBack`: callback with ImageData representation of the captured image. `errorCallBack`: callback with the error message to be invoked when something goes wrong.
forceTake Picture	successCallBack errorCallBack	Captures document while ignoring capture criteria. `successCallBack`: callback with ImageData representation of the captured image. `errorCallBack`: callback with the error message to be invoked when something goes wrong.
stopCapture	successCallBack errorCallBack	Stops the capturing of images (works both in single capture and continuous capture). `successCallBack`: callback with no data. `errorCallBack`: callback with the error message to be invoked when something goes wrong.
choose PictureAs Base64	successCallBack errorCallBack	Allows picture to be chosen from device photo library/gallery OR from device camera. This method returns selected Image as base64, hence best suited for the usecases where application picks images from gallery and send it for extraction. successCallBack: callback with Base64 data of captured or picked image errorCallBack: callback with error message to be invoked when something goes wrong. **Note** It is recommended to call/bind this method in some button click events instead of jquery page events or window load events to get full support from most of the browsers. This method works in manual mode: i.e useVideoStream is 'false'.
destroy	None	Cleans up internal the resources allocated by the create API call. Capturing must be stopped by the stopCapture API call before using destroy.

Example Code Snippet

```
//Initialize Capture singleton to work with video capturing
KfxWebSDK.Capture.create({
 useVideoStream: true,
 containerId: 'ID_CAMERA_DIV',
 preview: false
}, function() {
 console.info(‘Done’);
},
function(e) {
 console.info(e);
});

//Invokes method 'takePicture' on the singleton
KfxWebSDK.Capture.takePicture(function(imagedata)
  { // Do something with image data here }, function(e) { console.info(e);});
```

KfxWebSDK.SelfieCapture (Kofax Selfie Capture)

This class provides methods to take a selfie from either from the native camera or HTML5 Capture. It enhances the user experience by adding feedback while the user takes a selfie. This guidance makes it easier to take high quality selfies.

Native

Package name: com.kofax.selfiecapture

Global Namespace: KfxWebSDK

Class Name: SelfieCapture

JavaScript Closure

KfxWebSDK.SelfieCapture

APIs

API	Parameters	Description
loadModels	successCallBack errorCallBack	This method loads the required model files for `opencv` to detect face and eyes, respectively. In this case, the `haarcascade_eye` and `lbpcascade_frontalface` XML files are used. Invoke this method before launching Selfie Capture

API	Parameters	Description
create	options successCallBack errorCallBack	Creates a Selfie control based on selected options. It always uses the front camera. You can check your device supports selfie capture or not by using the `supportsSelfieCapture` method. If your device supports selfie capture, invoke the `loadModels` method before calling this method, or it will return an error. • `Options.containerId`: Empty divId, where the application developer wants to see a selfie camera preview along with selfie capture guidance. The div container must exist and be empty, otherwise an error will be thrown. The application developer has to properly set the size and position of the div. The SDK does not check the size and position or any other container CSS properties, this is a developer responsibility. • `Options.preview`: Boolean value representing whether or not to review the captured selfie using the SDK review control. In case of FALSE, the developer needs to implement its own review functionality. This option effects only HTML5 selfie capture, when the captured selfie is from the native camera there is no review screen available. • `Options.videoStream`: Boolean representing to follow either the standard capture or selfie capture process. Various capture criteria options can be set here as well (see `setOptions` below). If you do not set any capture criteria options here, the default values will be used (see `getDefaultOptions` below).

API	Parameters	Description
setOptions	options successCallBack errorCallBack	Sets selfie capture criteria. `{ containerId: 'divId', videoStream: true, preview: false, frameAspectRatio: 0, framePadding: 10, frameThickness: 10, frameColor: '#FF0000', outOfFrameColor: '#FFFFFF', guidanceFrameTransparency: 0.5, showEdges: false, edgesColor: '#FFFF00', edgesWidth: 4, criteria: { minFaceSize: 0.30, captureTimeout: 1700, centerToleranceFraction: 0.15 }, lookAndFeel: { forceCapture: 10 } }`
getOptions	successCallback errorCallback	Returns current selfie control options for selfie capture criteria, selfie capture guidance messages and other configurable UI options. • `successCallBack`: Callback with JSON object representing selfie capture control options. • `errorCallBack`: Callback with error message to be invoked when something goes wrong.
getDefault Options	successCallBack errorCallBack	Returns default selfie capture control options for selfie capture criteria, selfie capture guidance messages, and other configurable UI options. • `successCallBack`: Callback with JSON object representing selfie capture control options. • `errorCallBack`: Callback with error message to be invoked when something goes wrong.

API	Parameters	Description
takeSelfie	successCallBack errorCallBack	Starts the Auto Capture process • `successCallBack`: Callback with ImageData representation of the captured selfie. • `errorCallBack`: Callback with the error message to be invoked when something goes wrong.
forceTakeSelfie	successCallBack errorCallBack	Takes selfie while ignoring selfie capture criteria. • `successCallBack`: Callback with ImageData representation of the captured selfie. • `errorCallBack`: Callback with the error message to be invoked when something goes wrong.
stopCapture	successCallBack errorCallBack	Stops the capturing of selfies. • `successCallBack`: Callback with no data. • `errorCallBack`: Callback with the error message to be invoked when something goes wrong.
destroy	None	Cleans up internal resources allocated by the create API call. Capturing must be stopped by the `stopCapture` API call before using destroy.

Example Code Snippet

```
//Initialize SelfieCapture singleton to work with video capturing
KfxWebSDK.SelfieCapture.loadModels(function(){
KfxWebSDK.SelfieCapture.create({
   videoStream: true,
   containerId: 'ID_CAMERA_DIV',
   preview: false
}, function() {
   console.info(‘Done’);
},
function(createError) {
   console.info(createError);
});
}),function(loadModelsError){
 console.info(loadModelsError);
});
//Invokes method 'takeSelfie' on the singleton
KfxWebSDK.SelfieCapture.takeSelfie(function(imagedata){
// Do something with image data here
},function(takeSelfieError) {
console.info(takeSelfieError);
});
```

KfxWebSDK.DocumentExtractor (Kofax DocumentExtractor)

This class provides methods to extract documents and return extracted fields for RTTI and KTA servers. General use cases such as identity card and check deposit are best supported by this.

Native

Global Namespace: KfxWebSDK

Class Name: DocumentExtractor

JavaScript Closure

KfxWebSDK.DocumentExtractor

This singleton class contains methods you should use to extract documents.

APIs

API	Parameters	Description
authenticateIDWithKtaServer();	options, successCallback, errorCallback	Authenticates and extracts given set of Images and returns authentication and extraction results as server response for KTA server. options.url: KTA server URL. options.images: array of image's Base 64 needed for extraction. For example: check front and back or check front only. options.serverParameters: JSON Object representing parameters needed for extraction, such as username, password, processIdentityName, sessionId. options.timingInfo: Boolean value. When set to true returns JSON Object holding the request completion time log. **Note** This API is deprecated from 3.3 release so it is recommended to use executeRequestOnKtaServer API .

API	Parameters	Description
extractionWithKtaServer	options, successCallBack, errorCallBack	Extracts a given set of images and returns timingInfo and a list of extracted field names and values for the KTA server. options.url: KTA server URL. options.images: array of image's Base 64 needed for extraction. For example: check front and back or check front only. options.serverParameters: JSON Object representing parameters needed for extraction, such as username, password, processIdentityName, sessionId. options.timingInfo: Boolean value. When set to true returns JSON Object holding the request completion time log. **Note** This API is deprecated from 3.3 release so it is recommended to use executeRequestOnKtaServer API .
extractionWithRttiServer	options, successCallBack, errorCallBack	Extracts given set of images and returns timingInfo and a list of extracted field names and values for the RTTI server. options.url: RTTI server URL. options.images: array of images Uint8Array format needed for validation and extraction. For example: ID front or back or both. options.serverParameters: JSON Object representing parameters like xIDType, processImage, xregion,xCropImage, and xImageResize required to extract and validate the images. options.timingInfo: Boolean value. When set to true returns JSON Object holding the request completion time log. **Note** This API is deprecated from 3.3 release so it is recommended to use performExtractionWithRttiServer API
cancelExtraction		Cancels the current active service request.

API	Parameters	Description
performExtractionWithRttiServer	options, successCallback, errorCallback	Extracts given set of images and returns timingInfo and a list of extracted field names and values for the RTTI server. options.url: RTTI server URL. options.images: array of images Uint8Array format needed for validation and extraction. For example: ID front or back or both. options.serverParameters: JSON Object representing parameters like xIDType, xregion , processImage, xCropImage and xImageResize required to extract and validate the images. options.timingInfo: Boolean value. When set to true returns JSON Object holding the request completion time log.
executeRequestOnKtaServer	options, successCallback, errorCallback	Performs given request on KTA server. These requests includes Extraction, Authentication and Selfie Verification. Server parameters may vary based on the request. Refer to MobileID Documentation 2.x for more details about server parameters. Server response also may vary based on the request, need to parse the response based on that. options.url: KTA server URL. options.images: array of image's Base 64 needed for extraction. For example: check front and back or check front only. options.serverParameters: JSON Object representing parameters needed for extraction, such as username, password, processIdentityName, sessionId,transactionId. options.timingInfo: Boolean value. When set to true returns JSON Object holding the request completion time log.
loginToKTAServer	options, successCallback, errorCallback	Login to KTA server and returns a session id in case of success or else returns an error. options.url: KTA server URL options.username: username which is used to login to KTA server. options.password: password which is used to login to KTA server

Example code snippet

```
// Send data in options with RTTI server section
```

```
KfxWebSDK.DocumentExtractor.performExtractionWithRttiServer(options,successCallback,
errorCallback);
```

KfxWebSDK.ReviewControl (Kofax ReviewControl)

The Review Control has APIs used to create a review screen with Accept and Retake buttons.

This can optionally be used by the developer to manage the reviewing process. The ReviewControl is also embedded in the Capture module and can be enabled by setting options.preview to TRUE.

Native

Global Namespace: KfxWebSDK

Class Name: ReviewControl

JavaScript Closure

KfxWebSDK.ReviewControl

This class contains methods you can use to create a review screen and set the accept - retake buttons handler.

APIs

API	Parameters	Description
ReviewControl	containerId	Creates a review screen entity with the canvas and toolbar with Accept & Retake buttons. containerId: divId, where the developer wants to see a review screen. The div container must exist, otherwise an error will be thrown. The developer has to properly set the size and position of the div. The SDK doesn't check size and position or any other container css properties; this is a developer responsibility. The div container can be either empty or not. If the container is not empty, the review control will hide all nested child elements until Accept or Retake is pressed. **Note** This method (constructor) just creates the entity and prepares html elements. The review screen is not shown after this call.
review	imageData, acceptCallBack, retakeCallBack	Show the review screen with the imageData and the toolbar with Accept & Retake buttons. ImageData: image to be reviewed. This image is expected to have valid dimensions. acceptCallBack: callback to be invoked when the user press accept button. retakeCallBack: callback to be invoked when the user press retake button.

Example code snippet

```
//Call to show review screen
var reviewControl = new KfxWebSDK.ReviewControl(containerId);
```

```
reviewControl.review(imageData, acceptCallback, retakeCallback);
```

KfxWebSDK image processor

This class provides methods to convert an image to crop, scale, and setDPI.

For scaling, use the following recommendations for specific document types:

- Identification documents: 1 megapixel
- Passport: 2 megapixels
- Credit card: 1 megapixel
- Full-page bills: 3.5 megapixels
- Coupon bills: 2.5 megapixels
- Checks: 1 megapixel

API	Parameters	Description
autoCrop	imageData options successCallBack errorCallBack	Crops, deskews (and performs rectangularization if needed) on the input image after detecting the edges of the document. `imageData`: the raw bytes of the image typically from a canvas, for example ex: context. getImageData(). `Options.type` - One of the following values: • KfxWebSDK.document.MOBILE_ID: 0 • KfxWebSDK.document.CHECK_DEPOSIT: 1 • KfxWebSDK.document.BILL_PAY: 2 For now, only the MOBILE_ID type is supported. Other types are reserved for possible future use and research. The Type option helps the edge detector choose optimal processing parameters and defines the aspect ratio of the original document. If a check or bill type is specified, the success callback will return the original input image. `successCallBack` : this would contain cropped image `errorCallBack`: this would contain the appropriate error messages **Note** This API is deprecated from the 3.4 release.

API	Parameters	Description
scale	imageData options successCallBack errorCallBack	Scales the input image as per the specified `scalemegapixel` value in the options. `imageData`: the raw bytes of the image typically from a canvas, for example: `context. getImageData()` `options`: a JSON object for `scaleMegapixels`. Currently only one option (`scaleMegapixels`) is supported. The image is scaled to the specified megapixel value (width * height) For example, `var options = {scaleMegapixels: 1.2}`. The `scaleMegapixels` value should generally be greater than zero and should be only used to downscale the image and not upscale. If the `scaleMegapixels` value higher than the input image size is given, the original image is returned without any scaling. No error is thrown in this case. `successCallBack`: this would contain the scaled image `errorCallBack`: this would contain the appropriate error messages
setDPI	imageData Options successCallBack errorCallBack	Update the dpi for an image. `imageData`: the raw bytes of the image typically from a canvas ex: context. getImageData() Object: This is a JSON object containing dpi. There is only one option 'dpi'. Options.dpi: dpi value which we want to update for an image. var options = { dpi: 200 } `successCallBack`: this would contain jpeg binary as dataurl `errorCallBack`: this would contain the appropriate error messages

Example code snippet

```
KfxWebSDK.ImageProcessor.autoCrop(image, {
        type: KfxWebSDK.document.MOBILE_ID
},
function(imageData) {
        // Do something with image data here
}, function(e) {
        console.info(e);
});
```

Target frame cropping

`ImageProcessor` was extended with a new frame pre-cropping functionality. The new frame pre-cropping will perform a preliminary crop of the image based on the target frame and can improve EVRS page detection by eliminating some background noise. This only works if images are captured with our capture experience.

Cropping happens during image processing, prior to EVRS page detection.

To enable crop to frame, set the `useTargetFrameCrop` property of `KfxWebSDK.Capture` options to "true".

Target Frame Cropping has the following limitations:

- Target frame cropping must be used only in auto capture mode where the target frame available.
- Depending on the frame configuration there may only be a small effect from cropping, or there will be no cropping at all.
- If the feature is enabled, it is the user's responsibility to keep the document inside the frame.
- If the feature is enabled, we suggest you do not use client's auto-cropping feature.

KfxWebSDK.Utilities (Kofax Utilities)

Utilities contains the API to check if Web capture is supported or not, depending on the browser type and device model. Developers can use it to decide what capture create options to use.

Native

Global Namespace: KfxWebSDK

Class Name: Utilities

JavaScript Closure

KfxWebSDK.Utilities

This singleton class contains the method you should use to check if web capture supported. The first call may be slower, but once the result is returned, it is cached and subsequent calls return the cached result.

APIs

API	Parameters	Description
supportsAutoCapture	successCallback errorCallBack	Checks browser and device model support for Web capture. This is useful for checking compatibility for the advanced document detection based capture experience. Based on this, the developer can configure capture experience options while creating a capture control. `successCallback`: empty callback indicating autocapture is supported `errorCallback`: empty callback indicating autocapture is not supported

Example code snippet

```
KfxWebSDK.Utilities.supportsAutoCapture(function(){
      //Support for advanced capture is available
},function(){
      doStandardCapture();
});
```

KfxWebSDK.AppStats (Kofax AppStats)

This class provides methods to record app stats data while using the KfxWebSDK to capture, process, and extract documents. It will record the capture events needed to calculate the average capture times. It

also records the process events needed to calculate the average process times, and includes the ability to record application defined session events.

This class also includes a feature that can log field change events in order to analyze the extraction accuracy.

Native

Global Namesapce: KfxWebSDK

Class Name: AppStats

JavaScript Closure

KfxWebSDK.AppStats

APIs

API	Parameters	Description
initAppStats	successCallBack errorCallBack	Initializes the `AppStatsObject` and `AppStats` internal objects to prepare them for recording app stats data. Creates an environment object.
startRecord	successCallBack errorCallBack	Starts (or continues) recording app statistics.
stopRecord	successCallBack errorCallBack	Stops recording app statistics.
beginSession	Options successCallBack errorCallBack	This method gives the application a means of recording an application-defined session. Each session is a grouping in which all subsequent `appStats` operations will be logged with the same `sessionKey`, until the next `endSession` call. The `Options` object has the following parameters: `{ sessionKey: 'UniqueID', category: 'BillPay' }`
endSession	Options successCallBack errorCallBack	This method tells `appStats` to stop the session. Subsequent logging calls will not include a `sessionKey` in the app stats data, until the next time `beginSession` is called again. The `Options` object has the following parameters: `{ success: 'boolean', message: 'message string' }`
isRecording		Returns the recording status of the `AppStats`

API	Parameters	Description
ExportAppStats	successCallBack errorCallBack	Exports the recorded app stats data as a JSON object to the app. `AppStats` data resides in memory until the app refreshes or there are unexpected crashes, since the recorded app stats data is not persistent. It is the app's responsibility to call this method frequently to securely save the data.
logSessionEvent	SessionEventType successCallBack errorCallBack	This method provides the application a means of recording an application-defined session event. The `Options` object has the following parameters: `{ sessionType: 'string', response: 'response string' }`
logFieldChangeEvent	Options successCallBack errorCallBack	This method provides the application a means of recording a field change event. The `Options` object has the following parameters: `{ IsValid:boolean, ErrorDescription:'string'; FormattingFailed:boolean; DocumentID:'string'; FieldName:'string'; OriginalValue:'string'; Confidence:number; ChangedValue:'string'; }`

Example code snippet

```
KfxWebSDK.AppStats.initAppStats(function(initSuccess){
    console.log("init app stats initSuccess:"+initSuccess);
   },function(initError){
    console.log("init app stats initError"+initError);
   });
```

Note Field change events should be recorded from the app by explicitly calling the `KfxWebSDK.AppStats.logFieldChangeEvent()` method.

JSON definitions

The following sections provide definitions and examples of the JSON data used by this API.

General response structure

Success server response example

```
{
```

```
  "extractionClass": "ID",
  "classificationResult": [{
    "class": "ID",
    "confidence": 1.0
  }],
  "fields": [{
      "name": "DocumentType",
      "text": "DL",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Country",
      "text": "United States",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "FirstName",
      "text": "LAURA",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "MiddleName",
      "text": "CHARLOTTE",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "LastName",
      "text": "WILSON",
```

```
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "IDNumber",
      "text": "B1257878",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "DateOfBirth",
      "text": "1975-11-21",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address",
      "text": "421 N RODEO DRIVE",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Gender",
      "text": "F",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
```

```
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "ZIP",
      "text": "90210",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "State",
      "text": "CA",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "City",
      "text": "BEVERLY HILLS",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "NameSuffix",
      "text": "",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address2",
      "text": "421 N RODEO DRIVE",
      "valid": true,
      "errorDescription": "",
```

```
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address3",
      "text": "",
      "valid": false,
      "errorDescription": "The field extraction was not certain.",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 0.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address4",
      "text": "",
      "valid": false,
      "errorDescription": "The field extraction was not certain.",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 0.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address5",
      "text": "",
      "valid": false,
      "errorDescription": "The field extraction was not certain.",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 0.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Address6",
      "text": "",
      "valid": false,
      "errorDescription": "The field extraction was not certain.",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 0.0,
      "formattingFailed": false,
      "fieldAlternatives": []
```

```
    },
    {
      "name": "Class",
      "text": "C",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "CountryShort",
      "text": "USA",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "ExpirationDate",
      "text": "2018-11-21",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Eyes",
      "text": "BRN",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Hair",
      "text": "BLK",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
```

```
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Height",
      "text": "5-04",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "IssueDate",
      "text": "2013-10-31",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Nationality",
      "text": "",
      "valid": false,
      "errorDescription": "The field extraction was not certain.",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 0.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "Weight",
      "text": "112 lb",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
```

```
      "name": "License",
      "text": "B1257878",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "IsBarcodeRead",
      "text": "True",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "IsOcrRead",
      "text": "True",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "IsIDVerified",
      "text": "True",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "DocumentVerificationConfidenceRating",
      "text": "97",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
```

```
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "DocumentState",
      "text": "California",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    },
    {
      "name": "ProductVersion",
      "text": "2.0.0.0.0.43",
      "valid": true,
      "errorDescription": "",
      "left": -1,
      "top": -1,
      "height": -1,
      "width": -1,
      "pageIndex": -1,
      "confidence": 1.0,
      "formattingFailed": false,
      "fieldAlternatives": []
    }
  ],
  "sessionKey": "0ec25ce8-6490-4d43-9727-6ccc81c6ed3e",
  "environmentId": null,
  "instanceId": null,
  "documentId": "dae4f031-4499-41db-b8ad-043d75c4a823",
  "words": null,
  "processedImages": null,
  "result": "Success",
  "errorType": "None",
  "errorDescription": null
}
```


Error server response example


```
{
"message": "An error occurred processing job 3f02e0bd-7555-4b43-a331-1eaf5a144fde.
 Pool worker error calling service. Usage count: 2",
 "exceptionMessage": "Pool worker error calling service. Usage count: 2",
 "exceptionType": "System.Exception",
 "stackTrace": "   at Kofax.MobileTransformation.ProcessPool.CallService[TResult]
(Func`2 func)\r\n
  at Kofax.MobileTranformation.#.#(ProcessPool , String[] , String , Dictionary`2 ,
 Dictionary`2 ,
  Dictionary`2 )\r\n   at
 Kofax.MobileTransformation.Service.Controllers.TransformationController.
  Process(HttpRequestMessage request, # requestTiming, # dataLogging, String
 projectMapping, String[]
  imageFilenames, Dictionary`2 appStatsIds, Dictionary`2 userOptions, String class,
```

```
  Dictionary`2 xValues)",
 "innerException": {
  "message": "An error has occurred.",
  "exceptionMessage": "Script execution has been stopped because of runtime error:
 \nLine 29, Offset 0,
  (&H80131500) State not Recognized",
  "exceptionType": "System.ServiceModel.FaultException",
  "stackTrace": "\r\nServer stack trace: \r\n   at
 System.ServiceModel.Channels.ServiceChannel.
HandleReply(ProxyOperationRuntime operation, ProxyRpc& rpc)\r\n   at
 System.ServiceModel.Channels.
ServiceChannel.Call(String action, Boolean oneway, ProxyOperationRuntime operation,
 Object[] ins,
Object[] outs, TimeSpan timeout)\r\n   at
 System.ServiceModel.Channels.ServiceChannelProxy.
InvokeService(IMethodCallMessage methodCall, ProxyOperationRuntime operation)\r\n   at
 System.
ServiceModel.Channels.ServiceChannelProxy.Invoke(IMessage message)\r\n\r\nException
 rethrown at [0]:
\r\n   at System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage
 reqMsg,
IMessage retMsg)\r\n   at System.Runtime.Remoting.Proxies.RealProxy.
PrivateInvoke(MessageData& msgData, Int32 type)\r\n   at Kofax.MobileTransformation.
IKtmService.Process(Dictionary`2 appStatsIds, String[] imageFiles, Dictionary`2
 userOptions,
String classificationClass, Dictionary`2 xValues)\r\n   at Kofax.MobileTranformation.#.
<>c__DisplayClass2.<ProcessImage>b__0(IKtmService service)\r\n   at
Kofax.MobileTransformation.ProcessPool.CallService[TResult](Func`2 func,
String workerId)\r\n   at Kofax.MobileTransformation.ProcessPool.CallService[TResult]
(Func`2 func)"
 }
}
```

Capture set options

The following JSON definitions consist of options to set for the Capture module.

```
{
    frameAspectRatio: 0.629,
    framePadding: 5,
    frameCornerHeight: 10,
    frameCornerWidth: 60,
    frameCornerColor: '#00FF00',
    outOfFrameTransparency: 0.5,
    showEdges: false,
    edgesColor: '#FFFF00',
    edgesWidth: 4,
    guidanceSize: 150,
    useTargetFrameCrop: false,
    criteria: {
        minFillFraction: 0.65,
        maxFillFraction: 1.8,
        longAxisThreshold: 85,
        shortAxisThreshold: 60,
        centerToleranceFraction: 0.19,
        captureTimeout: 1700,
        turnSkewAngleTolerance: 10,
        pitchThreshold: 15,
        rollThreshold: 15
    },
    lookAndFeel: {
```

```
        documentSample: 'http://example.com/images/document_sample.jpg',
        forceCapture: 10,
        gallery: true
    }
};
```

Extraction Options

```
executeRequestOnKtaServer method JSON is
{
  url:'KTA url',
  images: 'array of image's base64,
  serverParameters : {
          sessionId:"sessionId",
          processIdentityName: "MobileID"
        },
  timingInfo : false
  }

performExtractionWithRttiServer method JSON is
{
  url:'RTTI url',
  images: 'array of images's Uint8Array,
  serverParameters : {
          processImage:true,
          xIDType: "ID",
          xCropImage: false,
          xImageResize: "ID-1"
        },
  timingInfo : false
  }
```

On-boarding application

The HTML5 SDK includes an on-boarding sample app which demonstrates key features of the Kofax Web SDK and components like Mobile ID, Check Deposit, Passport, Bill Pay, Credit Card and New Account.

Capture

The user captures a document through this feature. The capturing of a document can be done via the HTML5 SDK camera, device camera or the user can select an already existing image from the photo library.

HTML5 SDK capture enhances the user experience by giving feedback while the user captures a document. This feedback requires browser WebRTC support.

Mobile ID

The Mobile ID component allows the user to select ID Cards (Driver license, ID, etc.). Once the document has been selected, it is processed and then sent for extraction. The ID card is extracted at the server end, and then user can have a look at the extraction results. If the ID has a barcode on its back, it will be captured as an image.

Internally the Mobile ID component is divided into two components:

- **ID Front**

  The user is allowed to capture only the front side of the ID.
- **ID Front and Back**

  The user is allowed to capture both the front and backside of the ID.

ID authentication

ID authentication and validation services are available as an integrated single-API Software as a Service (SaaS), or as individual service modules. It provides the following benefits:

- Typical twenty-second turnaround in terms of customer experience.
- Proprietary image integrity checks The overall authentication process consists of several integrated elements.

The overall authentication process consists of several integrated elements.

Image integrity

A number of automated filters detect various types of tampered IDs.

Data validation

Data validation takes the information extracted from the ID document and compares it to 3rd party, government, and proprietary data sources to confirm the level of data accuracy.

Identity authentication

The document is classified and then certain types of forensic analysis are performed. This process also checks if the ID document number is in the proper alphanumeric format.

Mobile ID supports all the regions which are supported by Kofax Mobile ID Capture 2.2.

Facial recognition

Facial recognition is performed to establish and validate that the person capturing the ID is the real owner of the ID.

The user will submit a selfie portrait image to be compared to the image photograph extracted from the users ID card. The maximum number of facial recognition attempts is 3. This value is not configurable.

Passport

The Passport component allows user to select passport and get corresponding extraction results.

Check deposit

The Check Deposit component allows the user to select checks and get corresponding extraction results.

Internally the Check Deposit component is divided into two components:

- **Check Front**

  The user can select only the front of the check. When only the front has been sent for extraction, then the extraction fields which are related to the back of the check (Endorsement Found field, Image Mismatch field, etc.) always fail.
- **Check Front and Back**

  The user can select both sides of the check.

Bill pay

The bill pay component allows the user to capture bills and get the corresponding extraction results.

Credit card

The credit card component allows the user to capture credit or debit card images and get the corresponding extraction results. Both embossed and non-embossed cards are supported.

New Account

The New Account component allows the user to select required documents and open an account. The demo application doesn't really create an account but just demonstrates the concept. The New Account component demonstrates two ways of capturing user data:

- Auto fill the required information by capturing documents.
  1. The user has to select either DL or Passport to get the personal details.
  2. The user has to select Bill pay for residency details.
  3. The user has to enter the pay amount either by selecting Check or Credit Card to open an account.
- Manually fill in the necessary information. User can also manually fill the personal information and has to repeat steps 2 and 3 above.

Support and limitations

The following sections describe the supported devices and various limitations.

Supported devices

To check whether or not advanced capture is supported in a specific device and browser, the API `supportsAutoCapture` must be called.

This call is asynchronous and checks if:

- The secure protocol HTTPS is used.
- The browser supports WebRTC.
- The device has proper auto focus hardware support.

- The device can provide at least FHD back camera resolution via WebRTC.

Unsupported devices for Advance Capture mode

The following devices have poor auto focus capability. Advance Capture mode has been disabled for these devices in all browsers:

- Asus ZenFone 2
- Asus ZenFone 2E
- Asus Zenfone Zoom
- Asus Zoom 3
- HTC One M8, M9
- LG G2, G3, G4, G5, G6, G7
- LG Optimus G Pro
- Motorola Moto G
- Motorola Moto X 2nd Gen
- Nexus 4 and 9
- OnePlus 6
- Samsung S2, S3, S4, S4 mini, S5, S6, Note 4, Galaxy Tab S
- Samsung S7
- Samsung Galaxy Note 3
- Sony Xperia Tablet Z
- Sony Xperia Z1 and Z1s

Note Although these devices cannot be used with Advanced Capture, the native camera can still be used for the other types of capture.

Unsupported devices for the Selfie Capture experience

The following devices do not support the Selfie Capture experience. The Selfie Capture experience is also unsupported for Android OS versions before 5.0.

- Asus Zenfone Zoom
- Motorola Moto X and X 2nd Gen
- MI 5X
- Nexus 4, 5, 6, 6P, and 9
- Redmi Note 4
- Samsung Galaxy Tab S
- Sony Xperia Z1

Supported browsers

KfxWebSDK is targeted for mobile webkit based browsers. HTML5 features/specifications are slowly being adopted by most browsers, however as of now none of the browsers support all HTML5 features. Hence the degree of KfxWebSDK support varies from browser to browser.

The method `{supportsAutoCapture}` will allow a developer to check for browser and device support. For the `{Create}` method, that means a developer can choose to use Advanced Capture a.k.a Capture experience for supported browsers, or the device's native camera for unsupported browsers. All KfxWebSDK methods will report an error when used with an unsupported browser. See the API Reference guide for details on how individual methods and their error handling.

KfxWebSDK officially supports Advanced Capture on Android Chrome browser with minimum version 47 and iOS Safari browser with minimum version 11 and all the other browsers support native capture.

Note If there is problem either in the browser platform or device, the Advanced capture won't work. ex: Samsung S7 Edge has focus issues, due to this advance capture doesn't work.

Note When designing applications, the developer has to provide code to handle using the browser back button.

Limitations in the SDK

1. As KfxWebSDK is part of the HTML5 framework, its support depends on underlying webkit HTML5 support and security permissions.
2. Choose gallery only is a limitation in both Android & iPhone.
3. Choose camera only is a limitation in iPhone.
4. HTML5 SDK is not comparable with the native SDK in the capture experience and image processing functionality. HTML5 SDK is limited by WebRTC capabilities and javascript language performance. See9 and 10 below with detailed recommendations
5. Developers must not rename minified SDK file KfxWebSDK.j
6. The supported browsers for iPhone and iPad which can load captured image(either from the gallery or native camera) into the image blob are iOS Safari 9.x and above in iPad and iOS Safari 9.x and above in iPhone. For other versions user will not see the preview of the captured image.
7. With Android devices, for the best HTML5 Web capture experience, we recommend using Chrome version 47 or later.
8. SDK Guidance Capture is only supported over an HTTPS connection, and then only with supported browsers and devices. Native Capture will work with both HTTP and HTTPS connections, however HTTPS is required for the capture experience on Android.
9. As a general rule, do not attempt to capture documents that have been placed on a surface with complex patterns, shapes, or colors. A plain, contrasting surface is recommended.
10. For best results with HTML5 Web capture, ensure that the background is simple and has a strong contrast with the document (for example white document on a black background). Also, there should be no glare and no shadows on the document itself.
11. HTML5 SDK is not equivalent to the native SDK in terms of recording appstats events. Due to HTML5 SDK limitations, only a limited set of data on environment details and certain other events are being recorded. Please note the following:
    - `OSName` property has "HTML5" in the HTML5 SDK appstats.
    - Only `userAgent` details are returned from HTML5 SDK, instead of `Device OS`, `Carrier`, `Memory`, `Device ID`, etc. from the browser APIs. Only the `Model` property is appended.

- `ImageID` is not available in HTML5 image objects. Consequently, `ProcessedImageID` and `SourceImageID` are not recorded.
- `Response` [under the session event object]: this is application specific when logging the session event; the application has to provide the response string, if any.
- `DocumentID` : only field change events will have a `DocumentID`, as RTTI/KTA server will return the `DocumentID` in the extraction response. `DocumentID`s for other Image events from the SDK are not available.
- No image object has a `Storage Path` when it is captured or selected.

12. A user must set the correct native camera mode since the HTML5 SDK uses whatever is currently selected. For example, if the user last used the front camera, that is what will be displayed in the SDK.
13. Currently, the SDK doesn't work with the Samsung S5 model (SAMSUNG-SM-G900A) and Chrome version 56.0.2924.87. After loading the SDK JS file, the browser becomes unresponsive due to an unknown low level browser issue.

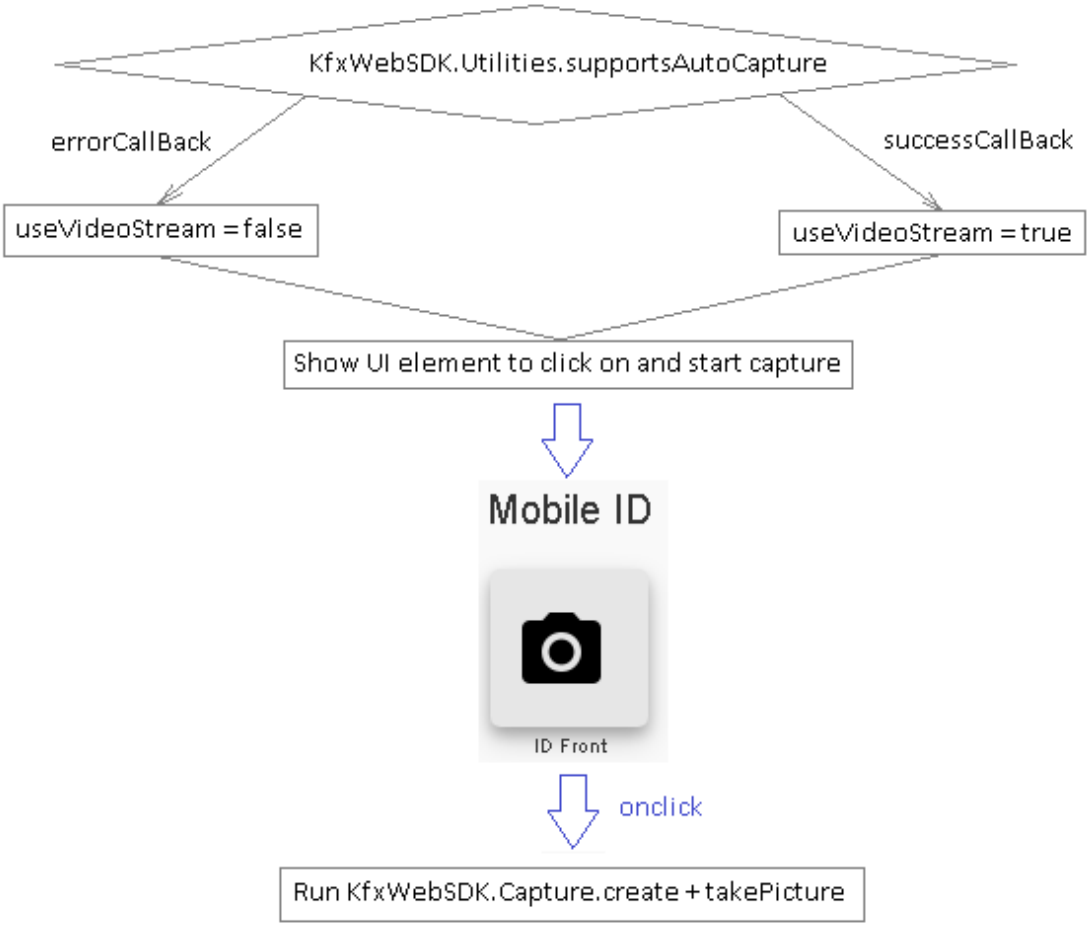
Implementation Limitations

- If you create `KfxWebSDK.Capture` with the option `useVideoStream = false` and then call `KfxWebSDK.Capture.takePicture(successCallBack, errorCallBack)` programmatically it won't work. The API runs the `input.click()` function internally. But this call has a major security restriction in browsers, it can be processed only if the call originated via the user's UI action. It will work from a click handler, for example, but it will be silently skipped without any error or exception in the browser console in the following cases:
  - Inside the `windows.onload` handler and all its subsequent functions.
  - Inside the `WebRTC getUserMedia` handler and all its subsequent functions.
  - Inside the `setTimeout` handler and all its subsequent functions.

  Error callback in `KfxWebSDK.Capture.create(options, successCallBack, errorCallBack)` in when `useVideoStream = true` originated in `getUserMedia`. The main disadvantage is that if we call "create API" with auto capture turned on (`useVideoStream = true`) and it fails, we can't directly recall it with `useVideoStream = false` and call `takePicture` to switch to the native camera or gallery. This is inconvenient and does not let a developer automatically switch from auto capture to standard capture mode.

  To address this problem we introduce `KfxWebSDK.Utilities.supportsAutoCapture(successCallBack, errorCallBack)` in release 3.2. This new API checks whether the browser and device support `WebRTC` by calling `getUserMedia` invoking a fake video element and stream.

  Our recommendation: check whether auto capture mode is supported before showing UI elements used to start capture. For example:



- . When we create `KfxWebSDK.Capture` with the option `useVideoStream = false` and then call `KfxWebSDK.Capture.takePicture(successCallBack, errorCallBack)` there may be 3 scenarios:
  - Callback `successCallBack` is called if the image is chosen successfully.
  - Callback `errorCallBack` is called if an error occurs.
  - Nothing is called if the user presses cancel or returns from the native camera or gallery application.

  The last scenario occurs because the native camera or gallery is launched as a standalone application not related to the browser and so any click on cancel or return (abort) cannot be tracked by browser and thus no event is fired. After the application is closed a user is simply taken to the last active rendered web page content.

  Our recommendation: before calling `takePicture`, first launch native video or gallery application be sure you have the desired web content you want the user to see if the application is cancelled.

Verifying captured images

Captured images should be reviewed and verified before sending them to the server. An image must have:

- A simple background with good contrast
- No cropping such that all four edges of the document are visible
- Minimal or no keystoning (perspective distortion)
- Readable text
- Minimal or no glare

There are three ways to implement such a review:

- Set the `KfxWebSDK.Capture` preview option to true (HTML5 SDK manages the review internally).
- Set the `KfxWebSDK.Capture` preview option to false and use the `KfxWebSDK.ReviewControl` class.
- Set `KfxWebSDK.Capture` preview option to false and implement your own review control.

Installation and hosting guide

An application can be hosted on any Web server, such as Apache or MAMP.

Overview

The HTML5 SDK can be used in your browsers to capture data received from mobile devices.

There are two ways to use the SDK in your Web application:

1. Create a new Web application.
2. Add an the existing Web application.

Creating a new Web application

After generating HTML files for your new Web application do the following:

1. Include the SDK CSS file (`KFXWebSDK.css`).
2. Include SDK Java Script file (`KFXWebSDK.js`)

   Note Do not move or rename anything in the SDK folder.

3. To ensure the SDK content is loaded successfully, or to debug any issue, use the Web Developer Tools and console. You can find this view in most popular browsers. You can also debug remotely on a device. Please follow the browser user guide to do it. For example, here is the link for Chrome, describing the remote debug process: `https://developer.chrome.com/devtools/docs/remote-debugging`.

Adding the SDK to an existing web application

Using the existing set of HTML files for the application:

1. Include the SDK css file (`KFXWebSDK.css`) in the correct location for your existing application.
2. Include SDK Java Script file (`KFXWebSDK.js`) in the correct location for your existing application.

   Note Do not move or rename anything in the SDK folder

3. To ensure the SDK content is loaded successfully, or to debug any issue, use the Web Developer Tools and console. You can find this view in most popular browsers. You can also debug remotely on a device. Please follow browser user guide to do it. For example, here is the link for Chrome describing remote debug process: `https://developer.chrome.com/devtools/docs/remote-debugging`.

Coding examples for HTML5 SDK

The following section provides code snippets for the HTML5 SDK. For details on the classes, methods, parameters, and so on, refer to the reference guide that ships with the product.

Initiate SDK capture with default options

```
var cameraOptions =  { containerId : "",
     preference : "camera",
   useVideoStream : true};

  KfxWebSDK.Capture.create(cameraOptions,function(createSuccess){
   KfxWebSDK.Capture.takePicture(function(imageData){

//success, user get the captured image in the ImageData  format .

   },function(error){

                       // error while taking the picture
   });
  },function(error){

// error while creating the capture control
  });
```

containerId

Specifies the DIV on which the camera will be launched.

preference

When advanced capture is turned off, you can choose between the gallery and the camera.

useVideoStream

A flag which allows the user to choose between advanced capture (HTML5 SDK camera) or standard capture (device camera or gallery).

Initiate SDK selfie capture with default options

```
var cameraOptions =  { containerId : "",
     preview : false,
   videoStream : true};

KfxWebSDK.SelfieCapture.loadModels(function(){
KfxWebSDK.SelfieCapture.create(cameraOptions,function() {
         KfxWebSDK.Capture.takePicture(function(imageData){
//success, user get the captured image in the ImageData  format .
     },function(error){
   // error while taking the selfie
   });

},
function(createError) {
   console.info(createError);
  // error while creating the selfie capture control

});
}),function(loadModelsError){
 console.info(loadModelsError);
});
```

containerId

Specifies the DIV on which the selfie camera will be launched.

preview

Boolean value representing whether or not to review the captured selfie using the SDK review control.

useVideoStream

A flag which allows the user to choose between HTML5 Selfie Capture or native camera.

Extraction

The extraction component extracts data from the captured document.

Code snippet:

```
var rttiOptions  = {

    url:"",
    images:[],
   timingInfo:true,
                            serverParameters:{}
                                                    };

ExtractionModule.extractionWithRttiServer(rttiOptions,function(result){

   // Success, The extraction results are obtained here .

     },function(errorMessage){

      // Error occurred while extraction

    });
```

Note For the URL and serverParameters fields, please refer to the appropriate Administrator's Guide. For example, for Mobile ID see the *Kofax Mobile ID Capture Administrator's Guide*.

Use HTML5 SDK as a node package

This section describes how to use HTML5 SDK as a node module using NPM, a JavaScript package manager. Instead of copying the HTML5 SDK into each application build folder manually, a node package can be made available to targeted users as part of a JavaScript library. Application developers can use the NPM command line interface (CLI) for installation and configuration tasks.

1. See if the node package already exists by going to the NPM website:

   `https://www.npmjs.com`

   If the package does not already exist, continue with the procedure.

   Important If the package already exists, errors will occur when publishing.

2. Create the package.json file in the HTML5 SDK root folder.

   The package.json file describes the node package name, version, and dependencies. You can create the file in either of the following ways:

   - In the NPM CLI, switch to the HTML5 SDK root folder and run the following command: `npm init`

     Note Make sure that the package does not already exist

   - Create the package.json file manually as in this example:

     ```
     {
       "name": "kfx-html5-plugin",
       "version": "1.0.0",
       "description": "A test plugin for HTML5 SDK",
       "main": "KfxWebSDK.js",
       "scripts": {
         "test": "echo \"Error: no test specified\" && exit 1"
       },
       "author": "Kofax",
       "license": "Kofax"
     }
     ```

3. In the KfxWebSDK.js file, add the following command below the last line:

   ```
   export default KfxWebSDK;
   ```

   This command exports objects from the node package so that application developers can access them from publically exposed APIs.

4. Create a user account on the NPM website (`https://www.npmjs.com`) to publish and use node packages.

   Users can log on by using the `npm init` command.