

# Mobile SDK

## Atalasoft MobileImage HTML5 SDK Developer's Guide

Version: 3.2.0

Date: 2017-05-23

©2013- 2017 Atalasoftware, 116 Pleasant St, Suite 321, Easthampton, MA 01027, U.S.A. All rights reserved.  
Use is subject to license terms.

Third-party software is copyrighted and licensed from Atalasoftware's suppliers.

THIS SOFTWARE CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF ATALASOFTWARE, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF KOFAX.

Atalasoftware, the Atalasoftware logo, and the Atalasoftware product names stated herein are trademarks or registered trademarks of Atalasoftware, Inc. in the U.S. and other countries. All other trademarks are the trademarks or registered trademarks of their respective owners. U.S. Government Rights Commercial software. Government users are subject to the Kofax. standard license agreement and applicable provisions of the FAR and its supplements.

You agree that you do not intend to and will not, directly or indirectly, export or transmit the Software or related documentation and technical data to any country to which such export or transmission is restricted by any applicable U.S. regulation or statute, without the prior written consent, if required, of the Bureau of Export Administration of the U.S. Department of Commerce, or such other governmental entity as may have jurisdiction over such export or transmission. You represent and warrant that you are not located in, under the control of, or a national or resident of any such country.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

# Table of Contents

<b>Preface</b> .....	<b>4</b>
Getting help for Kofax products.....	4
<b>Chapter 1: The HTML5 SDK</b> .....	<b>5</b>
Introduction.....	5
HTML5 extraction setup.....	5
Setting up the prerequisites.....	5
Using the HTML5 SDK with other HTML5 applications.....	6
HTML5 SDK external classes.....	6
KfxWebSDK.Capture (Atalasoft MobileImage).....	7
KfxWebSDK.ReviewControl (Atalasoft ReviewControl).....	9
KfxWebSDK image processor.....	10
KfxWebSDK.Utilities (Atalasoft Utilities).....	12
KfxWebSDK.AppStats (Atalasoft AppStats).....	13
JSON definitions.....	15
Capture set options.....	15
On-boarding application.....	15
Support and limitations.....	16
Verifying captured images.....	21
Coding example HTML5 SDK.....	21

# 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](http://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.

## Chapter 1

# The HTML5 SDK

## Introduction

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

## 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.

## Setting up the prerequisites

In addition to the above, 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 server is installed with the desired components configured and functioning.
2. Configure the server to allow cross-origin resource sharing (see [http://en.wikipedia.org/wiki/Cross-origin\\_resource\\_sharing](http://en.wikipedia.org/wiki/Cross-origin_resource_sharing)) for your HTML5 application
  - a. Open Internet Information Services (IIS) Manager.
  - b. Select the 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

In order to create a new HTML5 application and use/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 3<sup>rd</sup> 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.DocumentExtractor`
- `KfxWebSDK.ReviewControl`
- `KfxWebSDK.ImageProcessor`
- `KfxWebSDK.Utilities`
- `KfxWebSDK.AppStats` (Kofax AppStats)

The following sections describe these classes in detail.

## KfxWebSDK.Capture (Atalsoft MobileImage)

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	<p>Creates a Capture control based on given options. It will always use the rear camera.</p> <p><code>Options.containerId</code>: 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.</p> <p><code>Options.preference</code>: camera/gallery, from where the developer would like to capture a document.</p> <p><code>Options.preview</code>: 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.</p> <p><code>Options.videoStream</code>: Boolean representing to follow either the standard capture or document capture process.</p> <p>Various capture criteria options can be set here as well (see <code>setOptions</code> below). If you do not set any capture criteria options here, the default values will be used (see <code>getDefaultOptions</code> below).</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p><b>Note</b> The requirement to choose the gallery is a limitation in both Android and iPhone. Camera only is a limitation in iPhone.</p> </div>

API	Parameters	Description
setOptions	options successCallback errorCallback	<p>Sets various capture criteria.</p> <pre> {     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         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     } } </pre>
getOptions	successCallback errorCallback	<p>Returns current capture control options for capture criteria, capture guidance messages and other configurable ui options.</p> <p>successCallback: callback with JSON object representing capture control options.</p> <p>errorCallback: callback with error message to be invoked when something goes wrong.</p>
getDefault Options	successCallback errorCallback	<p>Returns default capture control options for capture criteria, capture guidance messages, and other configurable UI options.</p> <p>successCallback: callback with JSON object representing capture control options.</p> <p>errorCallback: callback with error message to be invoked when something goes wrong.</p>
takePicture	successCallback errorCallback	<p>Starts the Auto Capture process</p> <p>successCallback: callback with uint8 representation of the captured image.</p> <p>errorCallback: callback with the error message to be invoked when something goes wrong.</p>
takePicture Continually	successCallback errorCallback	<p>Starts Continuous Auto Capture process.</p> <p>successCallback: callback with uint8 representation of the captured image.</p> <p>errorCallback: callback with the error message to be invoked when something goes wrong.</p>



API	Parameters	Description
forceTake Picture	successCallback errorCallback	Captures document while ignoring capture criteria. successCallback: callback with uint8 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.
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() {}, function(e) { console.info(e);});
```

## KfxWebSDK.ReviewController (Atalasoftware 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.ReviewController

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	<p>Creates a review screen entity with the canvas and toolbar with Accept &amp; Retake buttons.</p> <p>Options.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.</p> <p><b>Note</b> This method (constructor) just creates the entity and prepares html elements. The review screen is not shown after this call.</p>
review	imageData, acceptCallback, retakeCallback	<p>Show the review screen with the imageData and the toolbar with Accept &amp; Retake buttons.</p> <p>ImageData: image to be reviewed. This image is expected to have valid dimensions.</p> <p>acceptCallback: callback to be invoked when the user press accept button.</p> <p>retakeCallback: callback to be invoked when the user press retake button.</p>

## Example code snippet

```
//Call to show review screen
var reviewControl = new KfxWebSDK.ReviewControl(options.containerId;
reviewControl.review(imageData, acceptCallback, retakeCallback);
```

## KfxWebSDK image processor

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

API	Parameters	Description
bitonal	imageData successCallback errorCallback	<p>Processes the image into a bitonal image.</p> <p>imageData: the raw bytes of the image typically from a canvas, for example: context.getImageData().</p> <p>successCallback: this would contain the resulting bitonal image.</p> <p>errorCallback: this would contain the appropriate error messages.</p> <p><b>Note</b> As of version 3.1 this has been deprecated.</p>

API	Parameters	Description
autoCrop	imageData options successCallBack errorCallback	<p>Crops, deskews (and performs rectangularization if needed) on the input image after detecting the edges of the document.</p> <p><code>imageData</code>: the raw bytes of the image typically from a canvas, for example <code>ex: context. getImageData()</code>.</p> <p><code>Options.type</code> - One of the following values:</p> <ul style="list-style-type: none"> <li>• <code>KfxWebSDK.document.MOBILE_ID: 0</code></li> <li>• <code>KfxWebSDK.document.CHECK_DEPOSIT: 1</code></li> <li>• <code>KfxWebSDK.document.BILL_PAY: 2</code></li> </ul> <p>For now, only the <code>MOBILE_ID</code> type is supported. Other types are reserved for possible future use and research.</p> <p>The <code>Type</code> 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.</p> <p><code>successCallBack</code> : this would contain cropped image</p> <p><code>errorCallBack</code>: this would contain the appropriate error messages</p>
scale	imageData options successCallBack errorCallback	<p>Scales the input image as per the specified <code>scaleMegapixel</code> value in the options.</p> <p><code>imageData</code>: the raw bytes of the image typically from a canvas, for example: <code>context. getImageData()</code></p> <p><code>options</code>: a JSON object for <code>scaleMegapixels</code>. Currently only one option (<code>scaleMegapixels</code>) is supported. The image is scaled to the specified megapixel value (<code>width * height</code>)</p> <p>For example, <code>var options = {scaleMegapixels: 1.2}</code>.</p> <p>The <code>scaleMegapixels</code> value should generally be greater than zero and should be only used to downscale the image and not upscale. If the <code>scaleMegapixels</code> value higher than the input image size is given, the original image is returned without any scaling. No error is thrown in this case.</p> <p><code>successCallBack</code>: this would contain the scaled image</p> <p><code>errorCallBack</code>: this would contain the appropriate error messages</p>
setDPI	imageData dpi successCallBack errorCallback	<p>Update the dpi for an image.</p> <p><code>imageData</code>: the raw bytes of the image typically from a canvas                      ex: <code>context. getImageData()</code></p> <p><code>dpi</code>: dpi value which we want to update for an image</p> <p><code>successCallBack</code>: this would contain jpeg binary as dataurl</p> <p><code>errorCallBack</code>: this would contain the appropriate error messages</p>

**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);
});
```

## KfxWebSDK.Utilities (Atalasoft 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
supportsAdvancedCapture (deprecated)	None	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.
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

```
if(!KfxWebSDK.Utilities.supportsAdvancedCapture()){
    doStandardCapture();
}else{
    $.mobile.navigate( "Capture.html",{transition:"none"} );
}
```

## KfxWebSDK.AppStats (Atalasoft 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

Package name : com.kofax.kmc.kut.utilities.appstats.AppStatistic

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' }

API	Parameters	Description
endSession	Options successCallback errorCallback	This method tells <code>appStats</code> to stop the session. Subsequent logging calls will not include a <code>sessionKey</code> in the app stats data, until the next time <code>beginSession</code> is called again. The <code>Options</code> object has the following parameters: { success: 'boolean', message: 'message string' }
isRecording	successCallback errorCallback	Returns the recording status of the <code>AppStats</code>
ExportAppStats	successCallback errorCallback	Exports the recorded app stats data as a JSON object to the app. <code>AppStats</code> 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 <code>Options</code> 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 <code>Options</code> 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.

### Capture set options

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

```
{
  frameAspectRatio: 0.628,
  framePadding: 5,
  frameCornerHeight: 15,
  frameCornerWidth: 70,
  frameCornerColor: '#00FF00',
  outOfFrameTransparency: 0.5,
  showEdges: false,
  edgesColor: '#FFFF00',
  edgesWidth: '4',

  criteria: {
    captureTimeout: 1700
    centerToleranceFraction: 0.15
    maxFillFraction: 1.8
    minFillFraction: 0.65
  },

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

## On-boarding application

The HTML5 SDK 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.

The following options can be set by the user:

- **Aspect ratio:**

The aspect ratio defines the rectangular frame that should match the document being searched. It defines aspect ratio used to draw the Static frame with respect to the effective camera area. To auto capture properly it is necessary to set this parameter according to the document aspect ratio. The following values should be set by application developers:

- Mobile ID 0.629
- Check 0.500
- Credit Card 0.623
- Passport 0.703

**Note** Do not use A4 documents with HTML5.

- **Padding:**  
Sets the minimum padding (in percent of length) to be maintained between the edges of the target frame and the image.
- **Demo Image:**  
The demo image acts as a sample image that can be captured.
- **Manual Capture Time:**  
After the set time (in seconds) has elapsed, the user is allowed to manually capture the image without regard to any capture constraints.
- **Use Video Frame:**  
A flag which allows the user to choose between advanced capture (HTML5 SDK camera) or standard capture (device camera or gallery).
- **Show edges:**  
When enabled shows the edges of the detected document.
- **Preference:**  
When advanced capture is turned off, you can choose between the gallery and the camera.

## 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 `supportAutoCapture` 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.



Following is a list of devices which have poor auto focus capability and so we disabled advance capture mode there in all browsers:

- Asus Zenfone Zoom (ASUS\_Z00XS)
- Asus ZenFone 2
- Asus ZenFone 2E
- Galaxy Note 3
- HTC One M8, M9
- LG G2, G3, G4
- Motorola Moto X 2nd Gen
- Nexus 9, Nexus 4
- Samsung S2 (GT-I9100), S3, S4, S4 mini, S5, S6, Note 4, Galaxy Tab S
- Samsung S7
- Sony Xperia Tablet Z
- Xperia Z1s
- Moto G

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

## 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 `{supportsAdvancedCapture}` 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/partially supported browsers. All KfxWebSDK methods will report an error when used with a partially supported browser. See the API Reference guide for details on how individual methods and their error handling.

Following is a list of browsers and which base versions KfxWebSDK are supported:

	Opera Mobile	iOS Safari	Android Browser	Firefox for Android	Chrome for Android
base version	30	8.0 and up	4.4.4	38	47 and up
Partial Support	NO	YES	YES(4.1, 4.2, 4.3)	YES	NO
Full Support	YES	NO	YES	NO	YES

**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. See 9 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 8.1 ,8.4 ,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. Server IPP profile limitations:

- Mobile IDs

The server IPP profile should always contain the following properties ( `_DocDimSmall_2.123_DocDimLarge_3.363`) which helps to resize images to 1009 \* 637 resolution for Acuant OCR. Without it some images may fail to be extracted. For example when cropping occurs on the client side, the IPP profile should be:

```
{
  "imageperfectionsettings": "_Do90DegreeRotation_4_DoScaleImageToDPI_300_DocDimSmall_2.123
  _DocDimLarge_3.363"
}
```

- Checks

HTML5 auto crop doesn't detect and align the MICR. It is highly recommended to use additional cropping on the server side which finds the MICR and can process checks more accurately. To provide better server-side cropping accuracy it is also recommended to not enable binarization on the client. The server IPP profile for both client and server-side image processing mode should be identical. Be sure it contains `DoCropCorrection` and `DoBinarization` settings.

- Passports

HTML5 auto crop simply crops as much image space from the passport as possible without rectangularization or deskew. This helps to remove unnecessary background in the image and

- send less data to the server, thereby reducing the output image size. The server IPP profile for both client and server-side image processing mode should be identical.
12. The HTML5 SDK web capture does not support guidance animation, which is supported in the native SDK.
  13. 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. `DocumentIDs` for other Image events from the SDK are not available.
    - No image object has a `Storage Path` when it is captured or selected.
  14. The HTML5 onboarding Display Preview function is not supported in iOS8.x.
  15. 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.
  16. Currently, the SDK doesn't work with the Samsung S5 model (SAMSUNG-SM-G900A) and the latest version of Chrome (56.0.2924.87). After loading the SDK JS file, the browser becomes unresponsive due to an unknown low level browser issue.
  17. Avoid setting HTML5 autocapture and cropping as default options in the HTML5 Onboarding App and the HTML5 sample app(deprecated in this release). HTML5 autocapture and cropping may negatively impact extraction accuracy.

## 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.

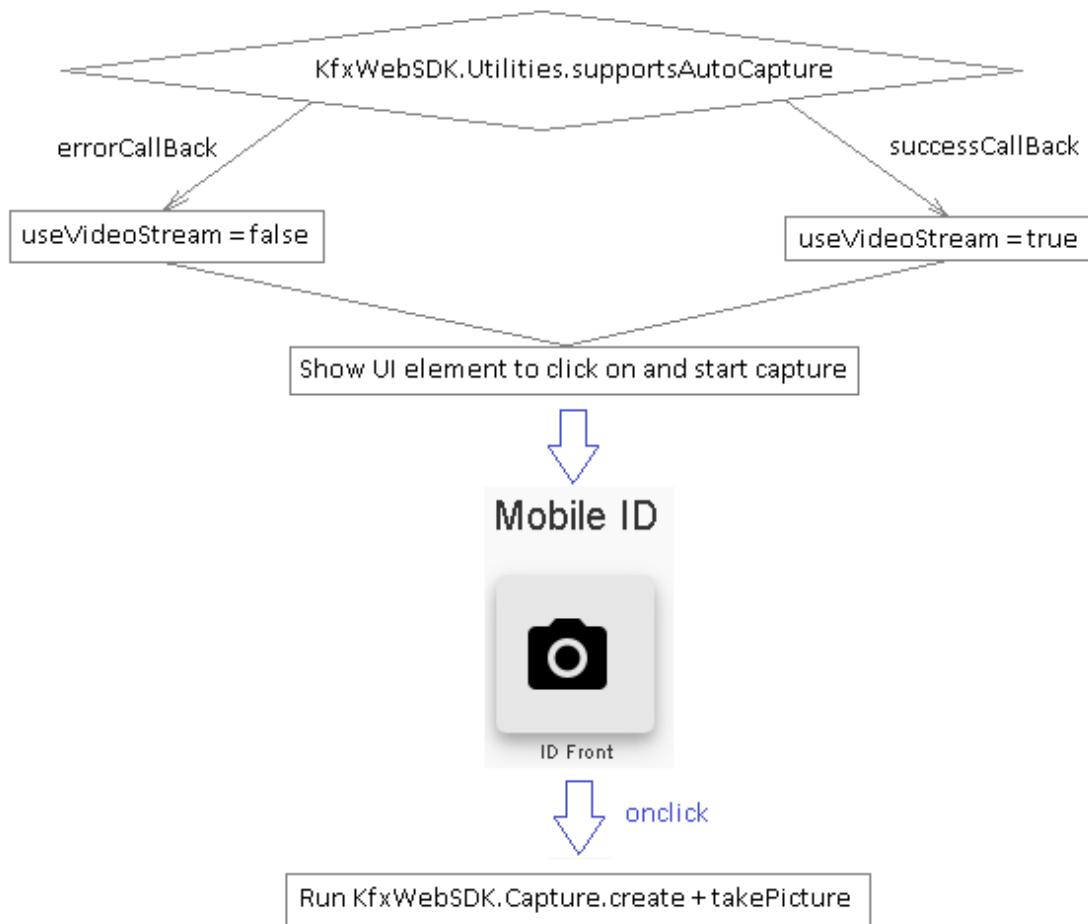
Error callback in `KfxWebSDK.Capture.create(options, successCallBack, errorCallback)` in when `useVideoStream = true` originated in `getUserMedia`. The main disadvantage is that if we call "create API" with auto capture turned on (`useVideoStream = true`) and it fails, we can't directly recall it with `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 keystone (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.ReviewController` class.
- Set `KfxWebSDK.Capture` preview option to false and implement your own review control.

## Coding example 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).