MiSnap™ SDK for Mobile Web

MiSnap 2.x for Mobile Web SDK

Developed by Mitek Systems, the MiSnap Mobile Web SDK enables users to capture images of optimal quality and usability for processing by Mitek's back-end solutions. MiSnap Mobile Web 2.x leverages the device's native camera application to acquire images for automatic analysis.

The following articles provide an overview about the MiSnap SDK, its features, and minimum system requirements required to help developers implement MiSnap technology using the MiSnap Software Development Kit (SDK).

2.5 & 2.6

SDK Overview

Developed by Mitek Systems, MiSnap Mobile Web enables users to capture images of optimal quality and usability for processing by the Mitek Imaging Platforms. MiSnap Mobile Web leverages device's native camera application to acquire images for automatic analysis.

Following articles provide an overview about MiSnap SDK, its features, and minimum system requirements required to help developers implement MiSnap technology using the MiSnap Software Development Kit (SDK).

What is MiSnap?

Mitek MiSnap™ is the mobile image capture component of all Mitek solutions. Built on patented, innovative science and advanced algorithms, MiSnap provides mobile device users a technically-superior, user-friendly method for capturing high-quality document images quickly and easily.

The MiSnap Software Development Kit (SDK) for Mobile Web provides mobile application developers with the tools and options needed to integrate their applications seamlessly with the MiSnap solution, while maintaining a customized user experience consistent with their unique brand image.

JAVASCRIPT API

By using our pure Javascript API, you can create any UI with any framework you like. This approach allows you to simply call methods in our Javascript API to process images and extract license data. Use this approach when you want to create your own UI. This approach allows complete control over the front-end / look and feel of your site. There are also a number of example HTML templates to develop various work-flows to get you up and running quickly.

SECURITY

Once the mobile imaging capture library (MiSnap) captures an image of a document, the image is reduced in size, compressed, and passed in RAM to the native application. MiSnap never communicates with the outside world itself and relies entirely on the customer’s application for communication.

PRODUCT HIGHLIGHTS

- Ability to capture an image using native camera or upload an image from a device
- Support on-device image quality analysis before sending an image to a server
- Support of webworkers
- Support capturing different documents such as driver license, ID cards, passport, checks, and generic documents etc
- Customizable user experience and branding
- Support both portrait and landscape modes based on document type

What's New in 2.5 / 2.6?

MiSnap SDK v2.6 for Mobile Web provides the following benefits to the customers:

- **New "Selfie" document type added**: Limited support added for selfie capture on mobile devices
- **Fixed issue with Chrome 72+**: Chrome introduced a new security feature in the latest update that conflicts with 2.5 This is fixed in v2.6.

MiSnap SDK v2.5 for Mobile Web provides the following benefits to the customers:

- **Enabled "Upload" use-case for desktop**: Tight coupling between the science and camera modules in MiSnap 2.4 limited the analysis and support of documents in Mobile web sdk 2.4 and prior to mobile devices only. In v2.5, we have refactored the sdk and separated the science and the camera modules, enabling developers to pass in an image of a document on web platforms and get instant quality (good or a bad image) feedback on-device for that document. With this, MiSnap mobile web 2.5 now **officially** supports the **upload** use-case for web platforms (mobile and **desktop**). **Note**: this version doesn’t support webcams.
- **Added new methods to call science directly**: Prior versions of the sdk provided single API for image analysis. 2.5 provides 2 additional APIs to the developers based on their feedback and would allow them to pass an image either as a blob or a data URL also, reducing the time to integrate the APIs. These new methods are exposed to allow you to call the science code directly by passing in image data:
    - MitekMobileWeb.processImageFile(params) - used to pass in image data from a file input
    - MitekMobileWeb.processDataURL(params) - used to pass in an image as a DataURL
    - MitekMobileWeb.processImageData(params) - used to pass in an image as an ImageData object

The response is identical to captureAndProcessImage. The new methods are described in more detail here.

- **Added minimum image-size quality check:** Image quality is key for successful document verification. For customers submitting uploaded images, we found that 3%-10% of the mobile verify transactions are rejected due to insufficient size of these documents. With this new check, sdk will provide instant feedback to the developers on-device if the size of the image passed does not match the minimum image standards. Developers can then provide useful feedback to their end-users, reducing wait time to get a similar response from the backend. To achieve best results a minimum image size was added to all science methods including the ones listed above. The minimum image size is 450px. If an image is passed in that does not have at least one side 450px long, it will return an error code of 13.
- **Ability to collect additional data for advanced analytics:** Currently, there is no way to categorize and track images submitted based on the web platforms (mobile, desktop) and based on the use-cases (upload, device-captured). Intelligent 2.5 APIs would automatically determine the platform (mobile or desktop) and will embed this information in the mibidata of the analyzed image, enabling customers to get the traffic insights for each platform. Same applies to use-cases depending upon the API invoked.
- **Ability to retrieve full image in addition to cropped image for the back of US DLs:** For the back of US DLs, v2.5 sdk will return the original image used for analysis in addition to the cropped barcode image. Developers are still supposed to send the cropped barcode to the Mitek backend, but the full image could be saved/used for their internal purposes depending on their business requirements.
- **Bug fixes:**
    - [Ticket 114082] **-** JS errors (TypeError) when integrating the sdk
    - [Ticket 113737, 113738] - Remove all references to XIP in the sdk and documentation

NOTE: This sdk should **not** be used to analyze images captured by **webcams** especially for the back of US DLs due to the low camera resolution of the webcams.

System Requirements

The MiSnap Mobile Web version 2.x SDK is supported on the following mobile platforms:

IOS

- Safari browser and iOS operating system version 7.1 and up.

ANDROID

- Chrome browser version 27.0 and up and Android operating system version 4.1 and up.
    - Devices should have at least a 4 mega-pixel camera.
    - Device / Chrome version / Android OS version tested list included with product documentation.

Release Notes

Mitek Systems continues to add features and make improvements to its software. This page describes "What's New in 2.x series", on which devices the SDK was tested, and any known and device-specific issues.

Change Log

SDK SIZE IN MB

	v2.6	v2.5	v2.4	v2.3
Manual Capture	0.75	0.75	0.52	1.1

ADDED IN 2.6.1

- Fixed an issue related to working with newer JS frameworks (e.g. React/Angular/etc.)
    - was causing the camera to not launch

ADDED IN 2.6

- Fixed an issue with a recent Chrome update (72+)
    - was causing the camera to not launch
- Added support for "Selfie" docType
    - limited support for capturing face (no IQA checks)

ADDED IN 2.5

- Added the following methods to allow calling our science code directly:
    - MitekMobileWeb.processImageFile(params);
    - MitekMobileWeb.processDataURL(params);
    - MitekMobileWeb.processImageData(params);
- Added a minimum image size restriction (450px)
- Return full image in addition to the cropped image for the back of the US DLs
- Ability to track if the image was captured on desktop or on mobile devices

Note: For detailed release notes, please click here.

FIXED IN 2.5

- [Ticket 114082] **-** JS errors (TypeError) when integrating the sdk
- [Ticket 113737, 113738] - Remove all references to XIP in the sdk and documentation

MODIFIED IN 2.4

- Enabled Four Corner and Glare detection for Passports
- Optimized thresholds for Checks and Documents
- Reduced size of mitekMobileWeb.js file from 880K to 520K, **41% reduction in the sdk size**.

Version 2.3

ID	Feature	Description
D-0812	Fixed image compression issue	Images are not getting compressed in Mobile Web 2.2 SDK when the web workers are disabled
I-01133	Updated SDK to not allow corrupted or non-image files	The SDK will not accept corrupted or non-image files as input
I-01137	EXIF data is retained after SDK image resize	The SDK now retains the EXIF data upon resizing the image (e.g. Orientation is retained)
I-01135	All hard-coded URL's removed from SDK	Hard-coded URL's referring to Mitek have been removed from all code
I-01134	SDK updated to remove all objects from global namespace	Webworker object (mitekWorker) removed from global namespace
B-05844	Fixed namespace collision issue within mitekMobileWeb.js file	Removed all global namespaced variables from files

Version 2.2

ID	Feature	Description
B-02144	Implemented webworker support for image processing	When enabled, webworkers are used to perform image processing without effecting the main UI javascript thread
B-03383	Implemented support for extracting data from Checks	Extracts data from checks (i.e. Routing, Account and Check numbers)

Version 2.1

ID	Feature	Description
B-02566	Implemented support for capturing Documents	Crops and binarizes images (e.g. used for trailing documents)
B-03359	Implemented support for extracting data from VIN documents.	Captures and extracts VIN data from Monroney stickers and Insurance Cards
B-02570	Implemented support for extracting data from Checks	Extracts data from checks (i.e. Routing, Account and Check numbers)
B-02346	Implemented support for extracting data from Passports	Extracts data from MRZ code on front of passport

Version 2.0

ID	Feature	Description	Benefit
B-02063	Implemented Four Corner Detection in Mobile Web for front images.	Detects four corners of license / rejects images that fail.	Higher success rates due to higher quality images being submitted to backend.
B-01963	Implemented Glare Detection in Mobile Web for front images.	Detects excessive Glare within the four corners of the image	Higher success rates due to higher quality images being submitted to backend.
B-01963	Implemented Exposure Detection in Mobile Web for front images.	Detects low exposure (images which are too dark)	Higher success rates due to higher quality images being submitted to backend.
B-01987	Implemented Focus Detection in Mobile Web for front images.	Detects images which are too blurry	Higher success rates due to higher quality images being submitted to backend.

Version 1.4.1

ID	Feature	Description	Benefit
B-01940	Implemented a workaround for a bug in iOS Safari that caused the browser to crash when taking a photo with the HTML5 camera API.	Changed the way how the browser handles the image as the camera is invoked.	iOS users can have a seamless capture experience without interruptions.

Version 1.4

ID	Feature	Description	Benefit
B-01382	Make MWeb 1.4 compatible with Mobile Verify 1.0.	MWeb 1.4 can call the Mobile Verify 1.0 endpoint for data extraction and license verification.	MWeb will function with the Photo Verify product.
B-01335	Make MWeb 1.4 compatible with Mobile Fill 2.0.	MWeb 1.4 will call the Mobile Fill 2.0 endpoint for data extraction.	MWeb will function with the latest version of Mobile Fill.
B-01347	Collect MibiData from Mobile Web and sends it to XIP.	MibiData records the end-user experience as they go through the MWeb process.	With MibiData stored in XIP, user experience can be viewed on the AdminUI for analysis or troubleshooting purposes.
US6307	Remove .svg images and replace them with .png.	.svg images require a specific MIME type in IIS for them to be displayed.	Removing these images reduces the friction for Mobile Web implementer.
US6213	Flexible implementation options for Mobile Web implementer.	Allows three implementation options for integration: everything Mitek provided, business logic only, or JavaScript API only.	Provide user more flexibility when implementing Mobile Web into their applications.
US6176	Update visualizations for Barcode capture and Front/Back error screens.	Used an animated GIF to guide users on how to capture the barcode. Included more distinctive images to notify users of errors.	New helpful visualizations for user guidance.

Version 1.3.1

ID	Feature	Description	Benefit
US6255	Optimize code minification.	Used UglifyJS to minify the Javascript.	Increase speed performance for CV processing.
US6225	New JPEG encoding with significant speed improvements.	Updated the MWeb API to use the canvas for image decoding.	Performance boost for both Front and Back CV processing.

Version 1.3

ID	Feature	Description	Benefit

US6020	Constructed Javascript API to be more suitable for external MWeb customers.	Refactor Mobile Web core to be pure Javascript API.	With the new Javascript API construction, MWeb customers can implement their own UI more easily.

Version 1.2.1

ID	Feature	Description	Benefit
DE1165	Removed the mocked screen that existed outside of the MWeb workflow experience.	The introduction screen is not a part of the MWeb workflow. Previously, going <*Back* redirected users to a mocked page that was cache.	Prevents users from navigating <*Back* to the introduction screen.

Version 1.2

ID	Feature	Description	Benefit
US5638	Expanded device and browser testing.	Broaden published test results to cover a wider array of device / chrome version / Android OS version combinations.	Customers who choose to use a whitelist approach will have enough test results to create a whitelist that covers a majority of the popular device/chrome combinations .

Version 1.1

ID	Feature	Description	Benefit
US5135	Refactor Mobile Web Configurations.	Refactor Mobile Web configuration to use an external configuration file for ease of implementation.	Makes the implementation of Mobile Web simpler.
US5236	Mobile Web: Chrome - Device coverage.	Mobile Web support is extended to include support for the Chrome browser.	Mobile Web compatibility for Android/Chrome users .
US5271	Mobile Web: Optimize Chrome UI.	Optimize usability for Android devices using the Chrome browser.	Fixed several issues via CSS including dynamic page & margin scaling, and proper button sizing.
US5328	Inclusion/exclusion code for Android devices.	Provided sample codes for devices inclusion and exclusion.	Specific Android devices can be blacklisted or whitelisted from Mobile Web based on model, Android OS, or Chrome version.

Devices Tested

The following is a list of devices were used for testing and certifying MobWeb 2.5 release.

Device	OS Version
LG L70	**4.4.2**
Samsung Galaxy Note 8	**7.1.1**
Samsung Galaxy S8	**7**
Moto G4	**6.0.1**
Nexus 9	**5**
iPad 3rd Generation	**7.1**
iPhone 6s	**11.2.5**
iPhone 6	**8.3**
iPhone 7 Plus	**10.0.2**
iPhone 8 Plus	**11.0.1**
iPhone SE	**10.3.3**

Tested on:

- MacBook Pro Safari 11.1/ Chrome 65.0.3325.181
- Windows 10 Chrome 66.0.3359.181
- Mobile browsers tested: Chrome and Safari

Known Issues

KNOWN ISSUES ON 2.X PRODUCT LINE:

- Front four corner detection is very sensitive to non-contrasting background and any break in the edges (user cannot hold front of license or have any glare on any edge)
- The Back of license wear and tear or print quality of barcodes will have an impact on ability to return results – specific licenses may have read errors which will require further diagnosis.
- If the application is configured to use the Back and Front if needed workflow and the back returns partial data, then only the Front capture data will be used. In future releases, the partial Back data will be merged with the Front capture data.
- DE-1121 Android/Chrome - certain device families save images onto Photo Gallery.
- DE-1191 Some Android/Chrome device/browser combos will hang during processing even though they are on the tested and supported device compatibility list – this is due to a low available memory situation. Clearing the browser data & cache will free up RAM/memory for the embedded CV processing.
- The Droid RAZR series of phones has an issue capturing images in Chrome. After capturing an image the message "Unfortunately, Camera has stopped" displays. A workaround is to use Firefox or another browser application on the device.
- With docType "Selfie", browsers may or may not open the front facing camera (browser support dependent)
- The camera fails to launch from the latest version of Chrome (72+) / this issue has been resolved in Mobile Web v2.6
- The camera fails to launch from newer JS frameworks (e.g. React/Angular/etc.) / this issue has been resolved in Mobile Web v2.6.1

Note: MiSnap mobile web sdk should **not** be used to analyze images captured by **webcams** especially for the back of US DLs due to the low camera resolution of the webcams.

Developer's Guide

The purpose of this section is to help developers implement MiSnap technology using the MiSnap Software Development Kit (SDK). This guide is intended for new as well as experienced users of MiSnap as follows—

- For developers new to MiSnap—this guide provides the information you need to set up, customize, and run the fully functional Sample App.
- For developers with existing implementations—this guide provides the information you need to upgrade MiSnap and use the latest version of the SDK.

MiSnap API

Processing Images from the SDK

There are two main ways to use the SDK. You can call "captureAndProcessImage" to launch the camera, process the image and call the science code to return IQA scores, or you can pass in a captured image and have the science code process it directly.

1 captureAndProcessImage This method allows you to launch the still camera, process the image and have the science code return IQA (Image Quality Assessment) scores. This api is typically used for document capture on mobile devices.

MitekMobileWeb.captureAndProcessImage(callback, imageType, captureParams, cvCallback);

API parameters:

- callback**:** the name of the function which is called to return data to
- imageType**:** type of image being captured. Valid values are “DL_Front”, “DL_Back”, "Document", "Check", "Passport", "Selfie" and "Other"
- captureParams (optional): override the default capture parameters. Examples of front and back captureParams can be located in the file "/js/MitekCaptureParams.js"
- cvCallback (optional): the name of the function which is called when an image is selected and the image is being processed on the client / used to show a message while image processing is occurring (e.g. "Please wait while we process your image.")

Note: The imageType of "Other" disables four corner detection and glare detection.

2 Passing an image to the science code directly There are 3 methods which allow you to pass an image to the science code to obtain IQA scores. This approach is typically used for **desktop upload use-case**.

MitekMobileWeb.processImageFile(params);

This method accepts blob as input. An example of this is displayed below.

```
<script>
function processResult(IQAResult) {
   console.log("IQAResult", IQAResult);
}

function processImageFile(fileData) {
    var params = {
        documentType: "DL_Front",
        imageData: fileData.files[0],
        callback: processResult
    }
    MitekMobileWeb.processImageFile(params);
}
</script>

<input type="file" accept="image/*" onchange="processImageFile(this)">
```

MitekMobileWeb.processDataURL(params);

This method accepts a DataURL as input. An example of this is displayed below.

```
<script>
function processResult(IQAResult) {
   console.log("IQAResult", IQAResult);
}

function processDataURL(dataURL) {
    var params = {
        documentType: "DL_Front",
        imageData: dataURL,
        callback: processResult
    }
    MitekMobileWeb.processDataURL(params);
}
</script>

<button onclick="processDataURL('/9j/4AAQ...')>Test DataURL</button>
```

MitekMobileWeb.processImageData(params);

This method accepts ImageData as input. An example of this is displayed below.

```
<script>
function processResult(IQAResult) {
   console.log("IQAResult", IQAResult);
}

function processImageData() {

    // get the ImageData from the Canvas
    var canvas = document.getElementById('canvas');
    var canvasContext = canvas.getContext('2d');
    var imageData = canvasContext.getImageData(0, 0, canvas.width,
canvas.height);

    var params = {
        documentType: "DL_Front",
        imageData: imageData,
        callback: processResult
    }
    MitekMobileWeb.processImageData(params);
}
</script>

<canvas id="canvas" width="1920" height="1440"/>
```

API parameter:

- documentType**:** type of image being captured. Valid values are “DL_Front”, “DL_Back”, "Document", "Check", "Passport" and "Other"
- callback**:** the name of the function which is called to return data to (IQA Result)
- imageData: varies depending on which method you call (e.g. Blob, DataURL, ImageData) / see example code above
- captureParams (optional): override the default capture parameters. Examples of front and back captureParams can be located in the file "/js/MitekCaptureParams.js"

```
var params: {
    documentType: "DL_Front",
    callback: callback, // the function to call to return the IQA
Scores/Result (JSON) to
    imageData: imageData, // varies depending on which method you call
(e.g. Blob, DataURL, ImageData);
    captureParams: {}  // (optional) override default values / see
section below
}
```

Note: The science code will return the same JSON response regardless of which method above you select.

Capture Parameters (Optional) for both the APIs:

```
var captureParams: {
    maxImageSize: 1920, // maximum size in pixels of one dimension of
the image (height or width)
    jpegEncodingQuality: 70, // jpeg compression to use for image (in
percentage / e.g. 70 = 70%)
    allowFileBrowser: false, // allows the user to optionally select an
image using the file browser on Android devices
    cvParams: {
      detectFourCorners: true,
      detectGlare: true,
      detectFocus: true,
      detectExposure: true,
    }
}

// capture the front of a license but override the default image
processing parameters
function captureFrontImage() {
   MitekMobileWeb.captureAndProcessImage(myCaptureFrontCallback,
'DL_Front', captureParams);
}
```

```
CaptureResult: {
    errors: [],   // array of errors
    status: "",   // result of CV processing (“success”, “failure”)
    Metrics: {},  // calculated IQA scores
    finalImage: null  // base 64 encoded image
    fullColorImage: null  // base 64 encoded full color image (note:
only applies to DL_Back)
}
```

Return (JSON Response/IQA Result)

Error Handling

If the status of a captureResult is “failure”, you will need to indicate to the client that the image failed to pass CV and cannot be processed by the back-end. The errors codes are returned in the CaptureResult if the status is “failure”.

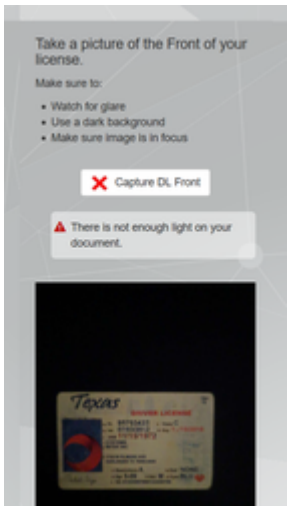
Below is a list of error codes that could be returned from a failed captureResult:

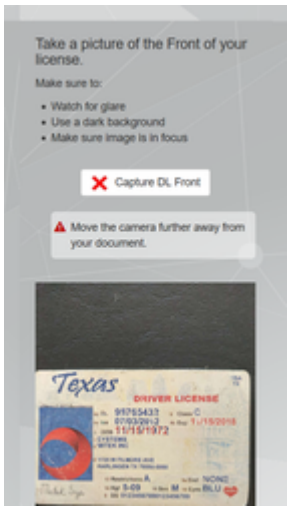
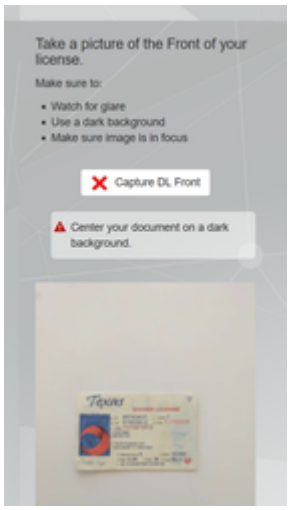
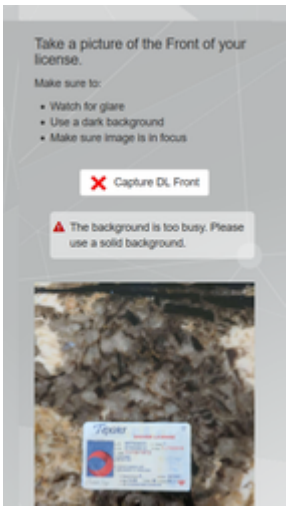
```
var MitekCaptureErrorCode = {
    GOOD: 0,    // Success
    NF: 1,      // Not found
    OOF: 2,     // Out of focus
    DARK: 3,    // Bad lighting
    MINFILL: 4, // Min fill
    SKEW: 5     // Skewed image
    MINWIDTH: 6, // Too far away (PDF417 barcodes only)
    GLARE: 7, // Glare detected
    FOURCORNERS: 8, // Four Corners not detected
    CORRUPTIMAGE: 9, // Image selected is corrupt
    MINPADDING: 10, // Not enough padding around document
    LOWCONTRAST: 11, // Low Contrast Background
    BUSYBACKGROUND: 12, // Busy Background
    IMAGETOOSMALL: 13 // Image too small (<450px)
}
```

Description of error codes and applicability for support documents:

SL No	Error Code	Error Message from Mitek Config	Applicable for document type DL_FRONT, DL_BACK ?	Applicable for documentType Check ?
1	NF	NF means barcode not found	Yes for DL_BACK only	No
2	OOF	The image is too blurry.	Yes	Yes
3	DARK	There is not enough light on your document.	Yes	Yes
4	MINFILL	Move the camera closer to your document. Document does not take up enough screen width.	No	No
5	SKEW	Hold camera directly over your document.	Yes for DL_BACK only	No
6	MINWIDTH	Move the camera closer to your document (PDF417 only)	No	No
7	GLARE	Too much glare on your image. Avoid bright lights.	Yes	Yes (Glare is on by default and can be disabled)
8	FOURCORNERS	We were unable to detect the four corners of your document.	Yes	Yes
9	CORRUPTIMAGE	Corrupted image provided	Yes	Yes
10	MINPADDING	Not enough padding around document. Move document further away.	Yes	Yes
11	LOWCONTRAST	Center your document on a dark background.	Yes, for DL_FRONT only	Yes
12	BUSYBACKGROUND	The background is too busy. Please use a solid background.	Yes, for DL_FRONT only	Yes
13	IMAGETOOSMALL	The image is too small.	Yes	Yes

***Mapping error codes to the UI messages** *added in SDK v2.4*

Wrong DL side	***Sharpness Check***	***Low-light Check***
		
Error code = 1	Error code = 2	Error code = 3
Insufficient Document Size*	***Skew Check****	***Glare Highlighted****
		
Error code = 4	Error code = 5	Error code = 7

Missing corners	***Invalid document***	***Insufficient padding****
		
Error code = 8	Error code = 9	Error code = 10
Low-contrast Background*	***Busy Background****	
		
Error code = 11	Error code = 12	

GETIMAGEWITHGLARE

This is a helper method to return an image (DataURL) with the Glare portion highlighted if a Glare error occurred. An example of this can be found in "/examples/front_only.html". See image above with Error Code 7 to see an example of an image with the glare highlighted.

```
MitekMobileWeb.getImageWithGlare(callback, captureResult,
glareBorderColor, glareBorderThickness);
```

callback:* the name of the function which is called to return image DataURL to*API parameters:

- captureResult**:** Javascript object returned from "captureAndProcessImage" method (see above)
- glareBorderColor (optional): The color of the border around the Glare (defaults to red)
- glareBorderThickness (optional): The thickness of the Glare border (defaults to 5px)

Code Example

```
// capture the front of a license but override the default image
processing parameters
function captureFrontImage() {
    MitekMobileWeb.captureAndProcessImage(myCaptureFrontCallback,
'DL_Front', captureParams);
}

function myCaptureFrontCallback(captureResult) {
    // if Glare was detected, show the image and highlight the Glare
    if (captureResult.errors.indexOf(MitekUtil.captureErrorCode.GLARE) >
-1) {
        MitekMobileWeb.getImageWithGlare(displayImageWithGlareCallback,
captureResult);
    }
}

// callback for: Glare was detected, display it with Glare highlighted
function displayImageWithGlareCallback(glareDataUrl) {
    document.getDocumentById('imagePreview').src = glareDataUrl;
}
```

If using a modern dev/build approach you can trying including the SDK files using webpack module loaders that can be found at NPM site. The below does work; however, results may vary depending on your overall approach i.e. build tool, environment, paths, etc;**Webpack ES6 Integration**

The SDK exposes a global: **MitekMobileWeb**

- worker-loader for mitekMobileWeb.js
- exports-loader for MitekMobileWebAPI.js, piexif.js, others

example (your file path may be different):

```
import MitekUtil from 'exports-loader?MitekUtil!./misnap/MitekAPIUtils'
import CV from 'exports-loader?CV!./misnap/cv'
import jsfeat from 'exports-loader?jsfeat!./misnap/jsfeat-min'
import piexif from './misnap/piexif'
import MitekMobileWeb from
'exports-loader?MitekMobileWeb!./misnap/MitekMobileWebAPI'
import MitekWorker from 'worker-loader!./misnap/mitekMobileWeb.worker'
class miSnap {
  init () {
    window.MitekMobileWeb = MitekMobileWeb
    window.MitekMobileWeb.mitekWorker = new MitekWorker()
  }
}
export default new miSnap()
```

Serve the SDK to browser over http:// or https:// using any application server (apache, node, python, etc;)Integrat

ion Testing

- Example: http://mylocalMachineVhost.com/dev/mitek/examples/
- Do not use => file://pathtosdk/js/etc; to view or work with SDK. If you do use browser File >> Open...the browser will not be in "web browser" mode and critical web feature are not enabled when viewing urls that start with => file:///

Code Snippets

For convenience, there are several HTML files which directly call the Mitek Mobile Web API to capture and extract data. Use these files as a point of reference to build out your UI. The files are located in “/examples/*.html”. Each file represents a different workflow.
The main workflows are:

- back_and_front_if_needed.html
  - Attempt to extract data from the back first and only prompt for the front if the back is unsuccessful or if the user does not have a barcode (note: uses Mobile Fill endpoint)
- front_and_back.html
  - Capture and extract the data from the front and back (note: uses Mobile Fill endpoint)
- back_only.html
  - Capture and extract the data from the back only (note: uses Mobile Fill endpoint)
- front_only.html
  - Capture and extract the data from the front and selected state only (note: uses Mobile Fill endpoint)
- mobile_verify.html
  - Capture and extract the data from the back barcode using Mobile Fill and then obtain a data comparison match score using Mobile Verify
- document.html
  - Capture a document image. (e.g. used for trailing documents)
- check.html
  - Capture a check image
- selfie.html
  - Capture a selfie
- direct_science.html
  - Provides examples on how to call the Mitek science code directly (e.g. passing in a dataURL to obtain IQA scores

Upgrade Guide

MIGRATING FROM 2.1.X TO 2.6 MOBILE WEB

If you have previously installed Mobile Web 2.1.x, this guide will describe the steps needed to change over to Mobile Web 2.4

The API for Mobile Web is backwards compatible with 2.1.x so no code changes should be needed to implement the new features of Mobile Web 2.5.

In order for Mobile Web 2.5 to work properly, two important code changes should be made:

1. make sure to set the global variable "mitekWorkerPath". This should be set to the path to the file "mitekMobileWeb.js". This ensures that web-workers are enabled. This will speed up the code and have the science work outside the main UI Javascript thread.
2. remove any <script> include tag to mitekMobileWeb.js. / Note: the worker will load this file on demand. Including it via a script tag could cause naming conflict issues.

To see the list of changes for Mobile Web 2.5/2.6, please see the Change Log.

Document Thresholds

DEFAULT MOBILE WEB THRESHOLDS

Parameter	Check	US DL / Global ID	Pas
Brightness [0-1000]	320	320	3
Sharpness [0-1000]	400	300	3
4 Corner [0-1000]	600	600	6
Glare [0-1000]	n/a	500	5
Image quality [0-100]	70	70	

UX Best Practices

USER EXPERIENCE BEST PRACTICES

We recommend to be creative in providing guidance on the following UX best practices to your users in your application: (for a passport, please picture the bio page only and not the full passport)

1. **ID must be placed on a dark surface.** Application works better when there is contrast between the ID and the surface on which it is placed. You might have to scan it multiple times if there is low contrast between the ID/passport and the surface. Preferably a plain black

paper or a black sheet with no text or printing on it.

2. **ID must be placed on a plain surface**. It should **not** be placed on a piece of a paper or a keyboard or a countertop or on a bed sheet or on your thigh. Holding the document in hand and scanning it might be rejected as well if key elements are obstructed. Use a plain background and ensure fingers are not obscuring the document.
3. **All 4 corners visible** When you image an ID/passport using the application, make sure to align the edges of the ID or passport bio page within the camera screen.
4. **Right distance between the ID and device:** Hold the device over the ID at a distance where the id card or passport bio page is completely inside the camera screen.
5. **No glare**. When you scan the document, please make sure there is no glare on it. You might have to scan it multiple times if there is glare.
6. **Avoid dark areas.** Scan the document in well lit areas and not in dark room or areas. Ideal is a normal day light or a well lit room.
7. **Hold steady.** Try to not to shake the device when you are taking the image of the ID document.

Mapping Error Messages to the UI Messages

As a UX best practice, we also recommend handling the error codes from the Mobile Web API and providing clear and meaningful action to the user. Click here for more examples.

Frequently Asked Questions (FAQ)

The topics below address some frequently asked questions (FAQs) to help you integrate the Mitek Mobile Web SDK.

WHICH VERSIONS OF ANDROID ARE SUPPORTED?

The minimum supported Android version is 4.1.

WHICH VERSIONS OF CHROME ARE SUPPORTED?

The minimum supported Chrome version is 47.0+.

THE JSON PAYLOAD IS FAIRLY LARGE. IS THERE A WAY TO REDUCE THIS?

There are two main settings you can override in the SDK. The "maxImageSize" and the "jpegEncodingQuality". Click on the link below to see an example of how to override the default settings (note: setting the "jpegEncodingQuality" below 50 is not recommended and could result in lower acceptance rates):

Sample Code to override default thresholds

I AM RECEIVING AN ERROR IN THE JAVASCRIPT CONSOLE, HOW DO I FIX IT?

```
MitekMobileWebAPI.js:8 Uncaught DOMException: Failed to construct
'Worker': Script at
'https://localhost/MitekMobileWebAPI/...ekMobileWeb.js' cannot be
accessed from origin 'https://localhost'.
    at Object.getWorker
(https://localhost/MitekMobileWebAPI/...bAPI.js:8:8287)
    at https://localhost/MitekMobileWebAPI/...bAPI.js:9:2011
```

There is a setting within the file "/js/MitekCaptureParams.js" that sets the path for the Mitek Webworker. The following line sets this path:

```
var mitekWorkerPath =
"https://localhost/MitekMobileWebAPI/src/mitekMobileWeb.js";
```

Optionally, you may also simply set the path for this global variable. Make sure that the path is accessible and points to the location of the

Webworker file "mitekMobileWeb.js". You will most likely need to substitute "localhost" with the domain name for your server as listed above. Also verify that if you are using a path using SSL / HTTPS that the website is also using SSL / HTTPS.

HOW DO I TURN OFF FOUR CORNER AND/OR GLARE DETECTION?

For certain documents, you may wish to turn off Four Corner and/or Glare detection. Below is a link that shows how to selectively turn on/off certain IQA checks:

Sample Code to override default thresholds

ARE ANY OPEN SOURCE LIBRARIES USED IN MISNAP?

MiSnap Mobile Web SDK uses the following open source libraries during image processing:

- cv.js - https://sourceforge.net/projects/opencvlibrary/
  - partial OpenCV port to Javascript
- js-feat.js - https://inspirit.github.io/jsfeat/
  - Computer vision library for image processing

MibiData

MibiData, *Mitek Business Intelligence Data*, is a proprietary analytics code that is embedded in the EXIF header of every image captured by MiSnap SDK. As MiSnap SDK is an offline solution, MiSnap team had no way to debug an issue reported by our customers. By capturing the user behavior and device specs (OS, model, and Camera specs etc) into an image captured on that device, we have been able to replicate the user behavior and improve the SDK over time.

This section describes what it is, its structure, its format, and provides a sample example to explain its structure.

Introduction

This article describes the content and format of Mitek's Business Intelligence Data (MibiData), which represents the data collected by MiSnap throughout the user's attempt to capture an image. The audience for this article are the members of IT departments interested in retrieving and processing this data from Mitek brand servers (e.g. Mobile Deposit servers or Mobile Imaging Platform servers) for the purposes of creating a general profile of the experience of using MiSnap.

OPERATION

MibiData is created at the completion of an end-user session. The static data, other than the User eXperience Payload (UXP), is collected at that time. The UXP, however, is continuously collected during the MiSnap lifetime and only stops once the user cancels or completes a snapshot.

MiSnap accumulates the user experience in real-time. Each video frame available is analyzed - some newer phones have a better frame rate than older phones and provide more measurements per second.

MiSnap assumes at first that it cannot detect a document, so it starts with a Not Found at time 0 measure. When the detection changes, i.e. when the document is found, another measure is captured; e.g., Closeness Fail, and that measure is added to the running UXP list. For example, if the user is slow to zoom in; many frames may capture a Closeness Fail measure. Rather than continuing to add the same issue to the list, no other information is added until a new issue occurs, e.g. when the user does move the phone close enough but the rotation angle is too large. Then, the Angle Fail measure is captured and added to the UXP list and MiSnap waits for another change to add to the running UXP list.

Dialog boxes will occasionally appear when the end user is unable to capture an image. When the dialog box appears, a begin measure is captured and added to the running UXP list, i.e. Timeout Dialog Begins (TB). After this measure disappears, e.g., the end user tapped a button; an ended measure is captured and added to the running UXP list, i.e. Timeout Dialog Ends (TE). These measures indicate the ending and beginning of attempts made by the end user to use MiSnap or manual modes.

Finally, when a MiSnap (MT) or Still Camera (ST) capture occurs, it is also added to the running UXP list, along with any end user cancellations (C). At this time, MiSnap is finished collecting the UXP data and prepares the JSON data for retrieval by a pull.

SIZE

The size of MibiData data sent to the server will vary slightly with each user experience attempting to capture an image. There will be some overhead to capture the device information contained in the payload, and some overhead to transport any parameters passed from the calling app. UXP data normally grows 14-18 bytes per measurement. The payload should seldom exceed 18K bytes per minute of user capture time. The maximum payload will be determined by a combination of the MiSnap parameters: CameraMaxTimeouts , CameraInitialTimeoutInSeconds and CameraTimeoutInSeconds. These settings may be provided by a Mitek server via settings from the server. If the server does not provide these settings, MiSnap falls back to internally hard-coded defaults.

The worst case scenario is a situation where the attempt to capture an image results in a good frame and then a bad frame in a repeating sequence, without an image being captured and considered eligible to be sent to the server. Assuming MiSnap is processing 15 frames per second on a high-end phone, MiSnap would process on average 320 bytes are captured every second or about 15K bytes in a minute. On average, a user will finish a capture within 10 seconds and average less than 10 messages a second, so the UXP should average about 1.5K bytes, and the entire BIP Payload should be about 2.5K bytes (or roughly 2.5% of a normal 100K byte JPEG). In most cases, a user will capture an image within 45 seconds, so such a maximum should never occur. (Note: for stress testing, a 28K byte BIP was uploaded to a Mitek server and the image survived with all EXIF data intact.)

MibiData Elements

MibiData Payload is a JSON Object as a string. The format is an array of the following optional key and value pairs as shown below.

`{“key”:”value”,...,”key”:”value”}`

Table 1 – Keys, Value Range, Units, and Description

Key	MiSnap 2.3	Value Range	Value Unit	Description
MibiVersion		String	text	MIBI data formatting version
UserAgent		String	text	Mobile Device's user agent string (contains data about the mobile device, browser, etc.)
MiSnapVersion		String	text	MiSnap version number from the library only (e.g. “2.3”)
MiSnapResultCode		String	text	Version of the add-on Mobile Web library (e.g. "SuccessStillCamera")
ImageSource.Platform		String	text	indicates platform / will be set to either "Mobile" or "Desktop"
ImageSource.Method		String	text	indicates the science method called
ImageSource.Upload		true/false	boolean	indicates if the image was uploaded
UXP		Array	JSON Array	The UXP payload - see the UXP table below

UXP

The format, where *n* is a decimal integer number representing milliseconds since MiSnap starts, is:

`[{“Label”:[n]}, {“Label”:[n,n]}, … {“Label”:[n]}]`

Table 2 contains the official Labels found in the UXP data, the parameter that the 1 or 2 byte character Label represents, the value range, the units of value, and a short description of each UXP event type.

Table 1 – UXP Object Keys, Value Ranges, and Descriptions

Parameter	Label	Value Range	Unit	Description
Image Captured	PC	[0..100000, 0..1000]	[msec, int]	Document captured by camera
Total IQA Check Time	TC	[0..100000, 0..1000]	[msec, int]	Total time to perform IQA checks on image
Total Encoding Time	EI	[0..100000, 0..1000]	[msec, int]	Time to encode the image
Encoding Quality	QX	0..100000	[msec]	Image encoding quality (note: default is 70%)
Image Width	QW	[0..100000, 0..1000]	[msec, int]	Image width
Image Height	QH	0..100000	[msec]	Image height
IQA Error Focus	QF	0..100000	[msec, int]	Captured image is too blurry
IQA Error Too Dark	QD	0..100000	[msec, int]	Captured image is too dark
IQA General Error	QQ	0..100000	[msec]	Captured image general failure
Image Capture Success	SU	0..100000	[msec]	Image captured successfully
Final Image Size	IS	0..100000	[msec]	Final image size
IQA Glare Failure	GD	0-100000, 0-1000	[ms, int]	Glare threshold succeeded
IQA Four Corner Failure	FC	0..100000	[msec]	Four corner failure / Unable to detect four corners of document

† Events may appear in UXP more than once per session.

MibiData Sample

MibiData, *Mitek Business Intelligence Data*, is a JSON Object (JavaScript Object Notation) that is embedded in the EXIF header of every image captured by MiSnap SDK. A full example of a MibiData is shown below:


```
{
  "MibiVersion": "1.4",
  "UserAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:53.0)
Gecko/20100101 Firefox/53.0",
  "Parameters": {
    "MiSnapVersion": "MobileWeb 2.6"
  },
  "MiSnapResultCode": "SuccessStillCamera",
  "ImageSource": {Platform: "Mobile", Method: "captureAndProcessImage",
Upload: "false" },
  "UXP": [
    {
      "PC": [
        2454,
        "DL_Front"
      ]
    },
    {
      "EI": [
        7273,
        "454"
      ]
    },
    {
      "QX": [
        7936,
        "0.7000000000000001"
      ]
    }
  ]
}
```