Acuant Hybrid SDK API Documentation

Last updated on – 08/25/2016

Smart from the start

acuantcorp.com

6167 Bristol Parkway Culver City, California 213.867.2625
Suite 330 90230

Contents

1 Introduction

The AcuantMobileSDK Plugin is a Cordova Plugin designed to simplify your development efforts. Processing of the captured images takes place via Acuant's Web Services. Acuant's Web Services offer fast data extraction and authentication with zero downtime.

Benefits:

- Process Enhancement: Faster data extraction and process images via Acuant's Web Services.
- Easy to set up and deploy.
- No maintenance and support: All maintenance and updates are done on Acuant servers.
- Secured Connection: Secured via SSL and HTTPS AES 256-bit encryption.

Acuant Web Services supports processing of drivers licenses, state IDs, other govt issued IDs, custom IDs, driver's license barcodes, passports, medical insurance cards etc. It also supports address verification, identity verification and personal verification.

For IDs from Asia, Australia, Europe, South America, Africa – we are support dd-mm-yyyy date format.
For IDs from Canada, USA – we are support mm-dd-yyyy date format.

For a complete list of regions, states, and countries supported for ID processing, please see Appendix F of ScanW document - http://www.id-reader.com/ftp/applications/sdk/docs/ScanW.pdf

To execute any Acuant Hybrid Mobile SDK method, a valid license key is required. Please contact sales@acuantcorp.com to obtain a license key.

This Acuant Hybrid Mobile SDK API documentation document has the detailed description of all the important functions a developer would need to write integration with Acuant Hybrid Mobile SDK.

Note: The Plugin will not modify the Status bar of the app.

2 Requirements

- Windows 10
- iOS 8.0 or later is required.
- AndroidSDK Version 17 or later.
- iPhone 4S and above.
- iPad 3 and above.
- iPad mini.
- iPod Touch 5G and above.
- 5 MP camera resolution or higher
- The card image must be taken in an acceptable light conditions to avoid glare and overhead lights for example.
- The card must preferably be fitted with in the brackets on the camera screen, to allow the picture to be taken at a maximum resolution.

NOTE: All methods require a success and failure callback. You can know which methods were called by checking if “id” equals to the name of the executed method.

Example:

```
var success = function (data) {
   log("success: " + data.id);
   if (typeof data === 'object') {
      if (data.id == 'mobileSDKWasValidated') {
         $("#progress_modal").toggleClass("hdn");
         $('#progress_modal').nsProgress('dismiss');
         if (data.data === true) {
            log('Framework validated');
         } else {
            log('Framework is not validated');
         }
      }
   }
}
```

3 Integration

Note : GitHub has recently changed the versioning for large files. To be able to download large files while cloning from GitHub repositories please make sure git-lfs is installed in the build machine. More information for git-lfs is available at https://git-lfs.github.com/. Please clone/update our SDK repository only after the git-lfs is installed.

A Add AcuantMobileSDK Plugin on each project

In order to add the plugin to your project, execute the following command line:

cordova plugin add < path to cordova-plugin-AcuantMobileSDK>

For example :

cordova plugin add Users/user/Desktop/AcuantHybridMobileSDK/cordova-plugin-AcuantHybridSDK

***Note** : In Windows please make sure the zcard.dll is present in "<Project folder>\platforms\windows\plugins\com.acuant.plugin.AcuantMobileSDK" in folder.Otherwise copy it from cordova-plugin-AcuantHybridSDK plugin folder.*

Also make sure the following cordova plugins are installed

- cordova-plugin-compat
- cordova-plugin-console
- cordova-plugin-dialogs
- cordova-plugin-network-information
- cordova-plugin-whitelist

For Windows platform the following plugin is also required.

- cordova-plugin-camera

4 Activate the license key

In order to activate the license key, use the following method:

```
AcuantMobileSDK.activateLicenseKey(successCallback, failureCallback, licenseKey);
```

Note: The license key only needs to be activated once. Execute this method only one time. Some licensees are issued by Acuant pre-activated and don't need further actions.

5 Initialize and create the SDK's instance

A Initialize with license key

In the below call, license key is validated and instance is created.

```
AcuantMobileSDK.initAcuantMobileSDK(successCallbalck, failureCallback, licenseKey, null);
```

Note: This method verifies if the license key is valid and it returns an instance that can be used to reference the methods. We recommend that you create one instance per session in order to optimize your resources.

B With license key and cloud address.

In the below call, license key is validated, the instance is created with the specified cloud address if you are hosting Acuant web services in your own data center. By default, iOS MobileSDK communicates with the Acuant data center.

```
AcuantMobileSDK.initAcuantMobileSDK(successCallbalck, failureCallback, licenseKey,
"cloud.myAddress.com");
```

The cloud Address must not contain “https://”
Ex: “https://cloud.myAddress.com/” must be written “cloud.myAddress.com”

Note: This method verifies if the license key is valid and it returns an instance that can be used to reference the methods. We recommend that you create one instance per session in order to optimize your resources.

C Check if the license key validation was successful or not.

In order to know if the license key validation has finished or to know if it was successful, catch the id type below on the success callback. This id is sent after the instance of the MobileSDK has been created.

```
var success = function (data) {
      if (data.id == 'mobileSDKWasValidated'){
      }
}
```

6 Capturing a card

A Add the card capture method.

In order to show the camera interface, set the card type (Medical Insurance, Driver’s License or Passport) as int value:

- Medical Insurance Card = 1
- Drivers License Card = 2
- Passport Card = 3

Depending upon the selected region and the variable state (isBackSide) it will show the correct camera interface.

For AcuantCardTypeMedicalInsuranceCard you can only use the manual capture interface.
For AcuantCardTypeDriversLicenseCard, depending on the region, you can only use the manual capture interface and the barcode capture interface.
For IDs from USA and Canada, use manual capture interface for the front side and use barcode capture or manual capture interface for backside.
For IDs from South America, Europe, Asia, Australia, Africa region use manual capture interface for both front and backside.
For AcuantCardTypePassportCard you can only use manual capture interface.

a Card capture interface with SDK initializations

In order to initialize the SDK and show the camera interface in the same step you must use the following method:

```
AcuantMobileSDK.initAcuantMobileSDKAndShowCardCaptureInterfaceInViewController(successCallback
, failure, licenseKey, cardType, region, isBackSide);
```

Note: if you are going to use any customization method, then you should create a previous instance of the SDK in order to set the camera customization.

b Manual Card capture interface without initialization

In order to call this function, you will need to initialize the SDK first and create an instance of the SDK to call the function (see point 5)

```
AcuantMobileSDK.showManualCameraInterfaceInViewController (successCallback, failure, cardType,
region, isBackSide);
```

c Barcode capture interface without initialization

In order to call this function, you will need to initialize the SDK first and create an instance of the SDK to call the function (see point 5)

```
AcuantMobileSDK.showBarcodeCameraInterfaceInViewController(successCallback, failure, cardType,
region, isBackSide);
```

d For all camera interfaces add the following methods to set the size of the card.

If the proper card size is not set, AcuantMobileSDK will not be able to process the card.

For Driver's License Cards

```
AcuantMobileSDK.setWidth(successCallback, failureCallback, 1250);
```

For Medical Insurance Cards

```
AcuantMobileSDK.setWidth(successCallback, failureCallback, 1012);
```

For Passport Documents

AcuantMobileSDK.setWidth(successCallback, failureCallback, 1478);

e Optional for all camera interfaces add the following methods to customize the appearance and final message on the camera screen.

Customize the initial message, default implementation says "Align and Tap" or “Tap to Focus”.

For iOS:
Use the following method to customize the text, duration, background color, position, and orientation

AcuantMobileSDK.setInitialMessage(successCallback, failureCallback, initialMessage, frameX, frameY, frameWidth, frameHeight, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, backgroundColorAlpha, duration, orientation);

For Android:
Use the following method to customize the text, duration, background color, position, and orientation

AcuantMobileSDK.setInitialMessage(successCallback, failureCallback, initialMessage, frameX, frameY, frameWidth, frameHeight, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, backgroundColorAlpha, duration, orientation);

Customize the capturing message, default implementation says "hold steady".

For iOS:
Use the following method to customize the text, duration, background color, position, and orientation

AcuantMobileSDK.setCapturingMessage(successCallback, failureCallback, capturingMessage, frameX, frameY, frameWidth, frameHeight, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, backgroundColorAlpha, duration, orientation);

For Android:
Use the following method to customize the text, duration, background color, position, and orientation

AcuantMobileSDK.setCapturingMessage(successCallback, failureCallback, capturingMessage, frameX, frameY, frameWidth, frameHeight, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, backgroundColorAlpha, duration, orientation);

NOTE:
Orientation values must be int 0 - 1 (Landscape - 0, Portrait – 1)
BackgroundColorRed values must be float 0 - 255
BackgroundColorGreen values must be float 0 - 255
BackgroundColorBlue values must be float 0 - 255

BackgroundColorAlpha values must be float 0 – 255

f Optional for all camera interfaces add the following methods to enable the flashlight and customize the flashlight button.

In order to enable the flashlight button (default customization).
AcuantMobileSDK.showFlashlightButton(successCallback, failure, showFlashlightButton);

In order to customize the flashlight button image.
AcuantMobileSDK.imageForFlashlightButton(successCallback, failure, flashlightButtonImageOn, flashlightButtonImageOff);

In order to customize the flashlight button frame.
AcuantMobileSDK.frameForFlashlightButton(successCallback, failure, frameX, frameY, frameWidth, frameHeight);

g Optional for barcode camera interfaces add the following methods to enable the barcode image cropping.

AcuantMobileSDK.setCanCropBarcode(success, failure, true);

h Optional method to enable the initial message on the barcode camera interface.
By default it is disabled.

AcuantMobileSDK.setCanShowMessage(success, failure, true);

i Optional method to pause the scanning of the barcode camera

AcuantMobileSDK.pauseScanningBarcodeCamera(success, failure);

j Optional method to resume the scanning of the barcode camera

AcuantMobileSDK.resumeScanningBarcodeCamera(success, failure);

k Optional method to show or not show the iPad brackets on the card capture interface

AcuantMobileSDK.showiPadBrackets(success, failure, true);

B Get the crop image from the success callback.
The cropped image returned came under base 64 format.

```
var success = function (data) {
   log("success: " + data.id);
   if (typeof data === 'object') {
      if (data.id == 'didCaptureCropImage') {
         $("#progress_modal").toggleClass("hdn");
         $('#progress_modal').nsProgress('dismiss');
         if (typeof data.data === 'string') {
            if (isFrontSide) {
               frontCardImage = data.data;
               isBarcodeSide = data.scanBackSide;
               var srcFront = "" + data.data;
               var imgFront = $('<img class="bordered" src="' + srcFront + '">');
               $("#front-image").empty();
               $("#front-image").prepend(imgFront);
               $("#front-image").removeClass("bordered");
               if (cardType == 2) {
                  navigator.notification.alert(
                     'Scan the backside of the license.',
                     showCameraInterfaceDLBack,
                     'AcuantHybridSampleSDK',
                     'OK'
                     );
               }
            } else {
               backCardImage = data.data;
               var srcBack = "" + data.data;
               var imgBack = $('<img class="bordered" src="' + srcBack + '">');
               $("#back-image").show();
               $("#back-image").empty();
               $("#back-image").prepend(imgBack);
               $("#back-image").removeClass("bordered");
            }
         }
      }
   }
};
```

C Get the original image from the success callback.

The original image returned came under base 64 format.

```
var success = function (data) {
   log("success: " + data.id);
```

```
    if (typeof data === 'object') {
        if (data.id == 'didCaptureOriginalImage') {
            if (typeof data.data === 'string') {
                originalImage = data.data;
            }
        }
    }
}
```

D Get the barcode data from the success callback.

In order to retrieve the cropped image captured by all card capture interface must use the following method:

```
var success = function (data) {
    log("success: " + data.id);
    if (typeof data === 'object') {
        if (data.id == 'didCaptureData') {
            if (typeof data.data === 'string') {
                barcodeStringData = data.data;
            }
        }
    }
}
```

7 Processing a card

A Add the card processing method.

processCardImage(successCallback, failureCallback, frontImage, backImage, barcodeStringData, autoDetectState, stateID, reformatImage, reformatImageColor, DPI, cropImage, faceDetection, signatureDetection, region, imageSource);

a For Driver's License Cards

In order to setup Driver License Card, set the following values.

AcuantMobileSDK.processCardImage(success, failure, frontImage, backImage, barcodeStringData, autoDetectState, stateID, reformatImage, reformatImageColor, DPI, cropImage, faceDetection, signatureDetection, region, imageSource);

Eg:

AcuantMobileSDK.processCardImage(success, failure, 2, backCardImage, barcodeStringData, true, -1, true, 0, 150, false, true, true, 0, 101);

Explanation of the parameters:

region - Integer parameter for the Region ID. Parameter value -
United States – 0
Australia – 4
Asia – 5
Canada – 1
America – 2
Europe – 3
Africa – 7
General Documents – 6

autoDetectState - Boolean value. True – SDK will auto detect the state of the ID. False – SDK wont auto detect the state of the ID and will use the value of ProcState integer.

stateID - Integer value of the state to which ID belongs to. If AutoDetectState is true, SDK automatically detects the state of the ID and stateID value is ignored. If AutoDetectState is false, SDK uses stateID integer value for processing. For a complete list of the different countries supported by the SDK and their different State integer values, please see Appendix F of ScanW document - http://www.id-reader.com/ftp/applications/sdk/docs/ScanW.pdf

faceDetection - Boolean value. True - Return face image. False – Won't return face image.

signatureDetection - Boolean value. True – Return signature image. False – Won't return signature image.

reformatImage - Boolean value. True – Return formatted processed image. False – Won't return formatted image. Values of ReformatImageColor and ReformatImageDpi will be ignored.

reformatImageColor - Integer value specifying the color value to reformat the image. Values –
Image same color – 0
Black and White – 1
Grayscale 256 – 2
Color 256 – 3
True color – 4
Enhanced Image – 5

DPI - Integer value up to 600. Reformats the image to the provided DPI value. Size of the image will depend on the DPI value. Lower value (150) is recommended to get a smaller image.

cropImage – Boolean value. When true, cloud will crop the RAW image. Boolean value. Since MobileSDK crops the image, leave this flag to false.

sourceImage – Define the source or type of image.
MobileSDK – 101

b For Medical Insurance Cards

In order to setup Medical Insurance Card, just set the following values.

AcuantMobileSDK.processCardImage(success, failure, frontImage, backImage, barcodeStringData, autoDetectState, stateID, reformatImage, reformatImageColor, DPI, cropImage, faceDetection, signatureDetection, region, imageSource);

Eg:

AcuantMobileSDK.processCardImage(success, failure, 1, backCardImage, barcodeStringData, true, -1, true, 0, 150, false, true, true, 0, 101);

Explanation of the parameters:

reformatImage - Boolean value. True – Return formatted processed image. False – Won't return formatted image. Values of ReformatImageColor and ReformatImageDpi will be ignored.

reformatImageColor - Integer value specifying the color value to reformat the image. Values –
Image same color – 0
Black and White – 1
Grayscale 256 – 2
Color 256 – 3
True color – 4
Enhanced Image – 5

DPI - Integer value up to 600. Reformats the image to the provided DPI value. Size of the image will depend on the DPI value. Lower value (150) is recommended to get a smaller image.

cropImage – Boolean value. When true, cloud will crop the RAW image. Boolean value. Since MobileSDK crops the image, leave this flag to false.

c For Passport

In order to setup Passport Card, just set the following values.

AcuantMobileSDK.processCardImage(success, failure, frontImage, backImage, barcodeStringData, autoDetectState, stateID, reformatImage, reformatImageColor, DPI, cropImage, faceDetection, signatureDetection, region, imageSource);

Eg:

AcuantMobileSDK.processCardImage(success, failure, 3, backCardImage, barcodeStringData, true, -1, true, 0, 150, false, true, true, 0, 101);

Explanation of the parameters:

faceDetection - Boolean value. True - Return face image. False – Won't return face image.

signatureDetection - Boolean value. True – Return signature image. False – Won't return signature image.

reformatImage - Boolean value. True – Return formatted processed image. False – Won't return formatted image. Values of ReformatImageColor and ReformatImageDpi will be ignored.

reformatImageColor - Integer value specifying the color value to reformat the image. Values –
Image same color – 0
Black and White – 1
Grayscale 256 – 2
Color 256 – 3
True color – 4
Enhanced Image – 5

DPI - Integer value up to 600. Reformats the image to the provided DPI value. Size of the image will depend on the DPI value. Lower value (150) is recommended to get a smaller image.

cropImage – Boolean value. When true, cloud will crop the RAW image. Boolean value. Since MobileSDK crops the image, leave this flag to false.

B Finally, check the callback process asking if "data.id" is equal to "`didFinishProcessingCardWithResult`".

a For Driver's License Cards

If using the Driver's License Card, add the following code:

```
if (cardType == 2) {
    frontCardImageResult = cardResult.licenceImage;
    backCardImageResult = cardResult.licenceImageTwo;
    faceImageResult = cardResult.faceImage;
    signatureImageResult = cardResult.signatureImage;
    resultString = "First Name -  " + cardResult.nameFirst + "</br>Middle Name -  " +
cardResult.nameMiddle + "</br>Last Name -  " + cardResult.nameLast + "</br>Name Suffix -  " +
cardResult.nameSuffix + "</br>ID -  " + cardResult.licenceId + "</br>License -  " + cardResult.license +
"</br>DOB Long -  " + cardResult.dateOfBirth4 + "</br>DOB Short -  " + cardResult.dateOfBirth +
"</br>Date Of Birth Local -  " + cardResult.dateOfBirthLocal + "</br>Issue Date Long -  " +
cardResult.issueDate4 + "</br>Issue Date Short -  " + cardResult.issueDate + "</br>Issue Date Local -  " +
cardResult.issueDateLocal + "</br>Expiration Date Long -  " + cardResult.expirationDate4 +
"</br>Expiration Date Short -  " + cardResult.expirationDate + "</br>Eye Color -  " + cardResult.eyeColor
+ "</br>Hair Color -  " + cardResult.hairColor + "</br>Height -  " + cardResult.height + "</br>Weight -  " +
cardResult.weight + "</br>Address -  " + cardResult.address + "</br>Address 2 -  " + cardResult.address2
+ "</br>Address 3 -  " + cardResult.address3 + "</br>Address 4 -  " + cardResult.address4 +
"</br>Address 5 -  " + cardResult.address5 + "</br>Address 6  -  " + cardResult.address6 + "</br>City -  "
+ cardResult.city + "</br>Zip -  " + cardResult.zip + "</br>State -  " + cardResult.state + "</br>County -  "
+ cardResult.county + "</br>Country Short -  " + cardResult.countryShort + "</br>Country Long -  " +
cardResult.idCountry + "</br>Class -  " + cardResult.licenceClass + "</br>Restriction -  " +
cardResult.restriction + "</br>Sex -  " + cardResult.sex + "</br>Audit -  " + cardResult.audit +
"</br>Endorsements -  " + cardResult.endorsements + "</br>Fee -  " + cardResult.fee + "</br>CSC -  " +
```

```
cardResult.CSC + "</br>SigNum -  " + cardResult.sigNum + "</br>Text1 -  " + cardResult.text1 +
"</br>Text2 -  " + cardResult.text2 + "</br>Text3 -  " + cardResult.text3 + "</br>Type -  " +
cardResult.type + "</br>Doc Type -  " + cardResult.docType + "</br>Father Name -  " +
cardResult.fatherName + "</br>Mother Name -  " + cardResult.motherName +
"</br>NameFirst_NonMRZ -  " + cardResult.nameFirst_NonMRZ + "</br>NameLast_NonMRZ -  " +
cardResult.nameLast_NonMRZ + "</br>NameLast1 -  " + cardResult.nameLast1 + "</br>NameLast2 -  " +
cardResult.nameLast2 + "</br>NameMiddle_NonMRZ -  " + cardResult.nameMiddle_NonMRZ +
"</br>NameSuffix_NonMRZ -  " + cardResult.nameSuffix_NonMRZ + "</br>Document Detected Name - "
+  cardResult.documentDetectedName + "</br>Nationality -  " + cardResult.nationality + "</br>Original -
" + cardResult.original + "</br>PlaceOfBirth -  " + cardResult.placeOfBirth + "</br>PlaceOfIssue -  " +
cardResult.placeOfIssue + "</br>Social Security -  " + cardResult.socialSecurity +
"</br>IsAddressCorrected -  " + cardResult.isAddressCorrected + "</br>IsAddressVerified -  " +
cardResult.isAddressVerified;
      if (cardRegion == 0 || cardRegion == 1){
         resultString = resultString + "</br>IsBarcodeRead -  " + cardResult.isBarcodeRead +
"</br>IsIDVerified -  " + cardResult.isIDVerified + "</br>IsOcrRead -  " + cardResult.isOcrRead;
      }
      resultString = resultString + "</br>Document Verification Confidence Rating -  " +
cardResult.documentVerificationRating;
}
```

b For Medical Insurance Cards

If using the Medical Insurance Card, add the following code:

```
if (cardType == 1) {
        frontCardImageResult = cardResult.reformattedImage;
        backCardImageResult = cardResult.reformattedImageTwo;
        resultString = "First Name - " + cardResult.firstName + "</br>Last Name -  " +
cardResult.lastName + "</br>Middle Name -  " + cardResult.middleName + "</br>MemberID -
" + cardResult.memberId + "</br>Group No. -  " + cardResult.groupNumber + "</br>Contract
Code -  " + cardResult.contractCode + "</br>Copay ER -  " + cardResult.copayEr +
"</br>Copay OV -  " + cardResult.copayOv + "</br>Copay SP -  " + cardResult.copaySp +
"</br>Copay UC -  " + cardResult.copayUc + "</br>Coverage -  " + cardResult.coverage +
"</br>Date of Birth -  " + cardResult.dateOfBirth + "</br>Deductible -  " +
cardResult.deductible + "</br>Effective Date -  " + cardResult.effectiveDate + "</br>Employer -
" + cardResult.employer + "</br>Expire Date -  " + cardResult.expirationDate + "</br>Group
Name -  " + cardResult.groupName + "</br>Issuer Number -  " + cardResult.issuerNumber +
"</br>Other -  " + cardResult.other + "</br>Payer ID -  " + cardResult.payerId + "</br>Plan
Admin -  " + cardResult.planAdmin + "</br>Plan Provider -  " + cardResult.planProvider +
"</br>Plan Type -  " + cardResult.planType + "</br>RX Bin -  " + cardResult.rxBin + "</br>RX
Group -  " + cardResult.rxGroup + "</br>RX ID -  " + cardResult.rxId + "</br>RX PCN -  " +
cardResult.rxPcn + "</br>Telephone -  " + cardResult.phoneNumber + "</br>Web -  " +
cardResult.webAddress + "</br>Email -  " + cardResult.email + "</br>Address -  " +
cardResult.fullAddress + "</br>City -  " + cardResult.city + "</br>Zip -  " + cardResult.zip +
"</br>State -  " + cardResult.state;    }
```

c For Passport Card

If using the Passport Card, add the following code:

```
if (cardType == 3) {
        frontCardImageResult = cardResult.passportImage;
        faceImageResult = cardResult.faceImage;
        signatureImageResult = cardResult.signImage;
        resultString = "First Name -  " + cardResult.nameFirst + "</br>Middle Name -  " +
cardResult.nameMiddle + "</br>Last Name -  " + cardResult.nameLast + "</br>Passport
Number -  " + cardResult.passportNumber + "</br>Personal Number -  " +
cardResult.personalNumber + "</br>Sex -  " + cardResult.sex + "</br>Country Long -  " +
cardResult.countryLong + "</br>Nationality Long -  " + cardResult.nationalityLong +
"</br>DOB Long -  " + cardResult.dateOfBirth4 + "</br>Issue Date Long -  " +
cardResult.issueDate4 + "</br>Expiration Date Long -  " + cardResult.expirationDate4 +
"</br>Place of Birth -  " + cardResult.end_POB;
    }
```

d For failure case.

If the image process fails for any reason you can then check the error in the failure callback by asking if error.id is equal to ‘didFailWithError’.

```
var failure = function (data) {
    if (data.id == ‘didFailWithError’) {
        log("Process error type:" + data.errorType + " Message: " + data.errorMessage);
        alert(data.errorMessage);
        $('#progress_modal').nsProgress('dismiss');
    } else if (data.errorType) {
        log("Error Type:" + data.errorType + " Message: " + data.errorMessage);
        alert(data.errorMessage);
        $('#progress_modal').nsProgress('dismiss');
    } else {
{
        log("failure: " + data.error);
    }
};
```

8 Error Types

AcuantErrorCouldNotReachServer = 0, //check internet connection
AcuantErrorUnableToAuthenticate = 1, //keyLicense are incorrect
AcuantErrorUnableToProcess = 2, //image received by the server was unreadable, take a new one
AcuantErrorInternalServerError = 3, //there was an error in our server, try again later
AcuantErrorUnknown = 4, //there was an error but we were unable to determine the reason,

try again later
AcuantErrorTimedOut = 5, //request timed out, may be because internet connection is too slow
AcuantErrorAutoDetectState = 6, //Error when try to detect the state
AcuantErrorWebResponse = 7, //the json was received by the server contain error
AcuantErrorUnableToCrop = 8, //the received image can't be cropped.
AcuantErrorInvalidLicenseKey = 9, //Is an invalid license key.
AcuantErrorInactiveLicenseKey = 10, //Is an inative license key.
AcuantErrorAccountDisabled = 11, //Is an account disabled.
AcuantErrorOnActiveLicenseKey = 12, //there was an error on activation key.
AcuantErrorValidatingLicensekey = 13, //The validation is still in process.
AcuantErrorCameraUnauthorized = 14, //The privacy settings are preventing us from accessing your camera.
AcuantErrorOpenCamera = 15 //There are an error when the camera is opened.

9 Supported Hybrid Frameworks

Acuant Hybrid Mobile SDK supports following hybrid frameworks:

Sencha
Phonegap
Intel XDK
ionic
Mobile Angular UI

10 Change Log

- Improved cropping in Android platform
- libpng version updated to resolve security vulnerabilities issue in Android platform
- Memory optimization to fix memory out of bound crashes in low memory devices
- Added DocumentDetectNameShort field for Android platform