Acuant iOS SDK API Documentation

Last updated on – 8/18/2015

Contents

1 Introduction

The AcuantMobileSDK.framework is a Cocoa Framework is designed to simplify your development efforts. The processing of the captured images takes place via Acuant's Web Services. Our Web Services offer fast data extraction and zero maintenance as software is looked after by Acuant on our optimized cloud infrastructure.

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 iOS Mobile SDK method, a valid license key is required. Please contact sales@acuantcorp.com to obtain a license key.

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

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

2 Requirements

- iOS 8.0 or later is required.
- iPhone 4 and above.
- iPad 3 and above.
- iPad mini.
- iPod Touch 5G and above.
- 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.

3 Integration

A Installation with CocoaPods

CocoaPods is a dependency manager for Objective-C, which automates and simplifies the process of using 3rd-party libraries like AcuantMobileSDK in your projects.

a Podfile

```
platform :ios, '8.1'
pod 'AcuantMobileSDK', :git => 'https://github.com/Acuant/AcuantiOSMobileSDKCocoaPods'
```

B Add AcuantMobileSDK.framework on each project

In order to add the framework to your project, drag the AcuantMobileSDK.framework folder into your project's file structure.

a Natives frameworks and libraries

Go to the target.
Click on “Build Phases”.
Expand “Link binary with libraries”.
Click on plus to add frameworks and libraries.
Add following frameworks.

- libc++.dylib.
- libiconv.dylib.
- AssetLibrary.framework
- SystemConfiguration.framework.
- AudioToolbox.framework
- AVFoundation.framework.
- CoreMedia.framework.
- CoreVideo.framework.
- CoreGraphics.framework
- QuartzCore.framework.
- libz.dylib.

b Targets

Go to the target.
Click on “Build Settings”.

b.1 Change following targets

Set “C Language Dialect” with GNU99
Set “C++ Language Dialect” with Compiler Default
Set “C++ Standard Library” with Compiler Default

b.2 Change following flags

Add on “PreProcessor” = CVLIB_IMG_NOCODEC
(GCC_PREPROCESSOR_DEFINITIONS = DEBUG=1 $(inherited)
CVLIB_IMG_NOCODEC)

C Add the import header in your appDelegate's header file.

```
#import <AcuantMobileSDK/AcuantMobileSDKController.h>
```

D Create and initialize the instance in your appDelegate's implementation file.

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

```
    //Obtain the main controller instance
    _instance = [AcuantMobileSDKController
initAcuantMobileSDKWithLicenseKey:@"MyLicensekey" andDelegate self];
```

b With license key and cloud address.
The cloud Address must not contain "https://"
Ex: "https://cloud.myAddress.com/" must be written "cloud.myAddress.com"
Note: Only set cloud address if you are hosting Acuant web services in your own data center. By default, iOS MobileSDK communicates with the Acuant data center.
In the below call, license key is validated and instance is created with the specified cloud address.

```
//Obtain the main controller instance
    _instance = [AcuantMobileSDKController
initAcuantMobileSDKWithLicenseKey:@"MyLicensekey" delegate self
andCloudAddress:@"cloud.myAddress.com"];
```

c If your instance was created previously

```
    //Obtain the main controller instance
   _instance = [AcuantMobileSDKController initAcuantMobileSDK];
```

4 Validating a license key

A To activate a license key.
In order to activate the license key, just set the license key and add the following code:

```
- (IBAction)activateAction:(id)sender {
    [_instance activateLicenseKey:_licenseKeyText.text];
}
```

B Optionally, in order to check if the license key validation was successful or not, use the method below.
In order to know if the license key validation has finished or to know if it was successful, use the method below. This method is called after the instance of the MobileSDK has been created.

```
-(void)mobileSDKWasValidated:(BOOL)wasValidated{
    _wasValidated = wasValidated;
}
```

5 Capturing a card

A SDK Configuration for card capture interface.

In order to show the camera interface, set the AcuantCardType (AcuantCardTypeMedicalInsuranceCard, AcuantCardTypeDriversLicenseCard, AcuantCardTypePassportCard).
Depending what you will select, 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 back side.
For IDs from South America, Europe, Asia, Australia, Africa region use manual capture interface for both front and back side.
For AcuantCardTypePassportCard you can only use the auto capture interface.

a In the header file where you'll be doing the parsing, add the following import.

```
#import <AcuantMobileSDK/AcuantMobileSDKController.h>
```

b In the same header file, implement the AcuantMobileSDKControllerCapturingDelegate.

```
@interface ISGViewController ()
<AcuantMobileSDKControllerCapturingDelegate,
AcuantMobileSDKControllerProcessingDelegate>
```

B Card capture interface methods.

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:

```
[AcuantMobileSDKController
initAcuantMobileSDKWithLicenseKey:licenseKey
AndShowCardCaptureInterfaceInViewController:self delegate:self
typeCard:_cardType region:_region isBarcodeSide:_isBarcodeSide];
```

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.
Ex:

```
_instance = [AcuantMobileSDKController initAcuantMobileSDK];
[_instance setWidth:1009];
 [AcuantMobileSDKController
initAcuantMobileSDKWithLicenseKey:licenseKey
AndShowCardCaptureInterfaceInViewController:self delegate:self
typeCard:_cardType region:_region isBarcodeSide:_isBarcodeSide];
```

b 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 4)

```
[_instance showCameraInterfaceInViewController:self delegate:self
cardType:_cardType region:_region isBarcodeSide:_isBarcodeSide];
```

c Methods to set the size of the card.

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

For Driver's License Cards

```
-(void)showCameraInterface{
    [_instance setWidth:1009];
```

For Medical Insurance Cards

```
-(void)showCameraInterface{
    [_instance setWidth:1009];
```

For Passport Documents

```
-(void)showCameraInterface{
    [_instance setWidth:1478];
```

d Optional 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”.

```
[_instance setInitialMessage:@"Initial Message" frame:CGRectMake(0, 0,
0, 0) backgroundColor:[UIColor redColor] duration:5.0
orientation:AcuantHUDLandscape ];
```

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

```
[_instance setCapturingMessage:@"Capturing Message"
frame:CGRectMake(0, 0, 0, 0) backgroundColor:[UIColor greenColor]
duration:5.0 orientation:AcuantHUDLandscape];
```

e Optional method to enable cropping of the barcode image.

By default it is disabled.

```
[_instance setCanCropBarcode:YES];
```

```
Note: The barcode cropped image will be received with the
didCaptureImage delegate method.
```

C AcuantMobileSDKControllerCapturingDelegate protocol to handle the capturing.

a Required delegate method

a.1 didCaptureImage

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

```
-(void)didCaptureImage:(UIImage *)cardImage
scanBackSide:(BOOL)scanBackSide{
    switch (_sideTouch) {
        case FrontSide:
            [_frontImage setImage:cardImage];
            break;
        case BackSide:
            [_backImage setImage:cardImage];
            break;
        default:
            break;
    }
    if (scanBackSide) {
        _sideTouch = BackSide;
        [UIAlertView
showSimpleAlertWithTitle:@"AcuantiOSMobileSDKSample" Message:@"Scan
the backside of the license."  FirstButton:ButtonOK SecondButton:nil
Delegate:self Tag:1];
    }
}
```

Note: For AcuantCardTypeMedicalInsuranceCard capturing backside is optional but for AcuantCardTypeDriverLicenseCard capturing back side is a must.

a.2 didCaptureData delegate method

In order to retrieve the barcode string by the barcode capture interface for AcuantCardTypeDriverLicenseCard you must use the following method:

```
-(void) didCaptureData:(NSString *)data{
    self.barcodeString = data;
}
```

a.3 didFailWithError delegate method

```
-(void)didFailWithError:(AcuantError *)error{
    NSString *message;
    switch (error.errorType) {
        case AcuantErrorTimedOut:
```

```
            message = error.errorMessage;
            break;
        case AcuantErrorUnknown:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToProcess:
            message = error.errorMessage;
            break;
        case AcuantErrorInternalServerError:
            message = error.errorMessage;
            break;
        case AcuantErrorCouldNotReachServer:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToAuthenticate:
            message = error.errorMessage;
            break;
        case AcuantErrorAutoDetectState:
            message = error.errorMessage;
            break;
        case AcuantErrorWebResponse:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToCrop:
            message = error.errorMessage;
            break;
        case AcuantErrorInvalidLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorInactiveLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorAccountDisabled:
            message = error.errorMessage;
            break;
        case AcuantErrorOnActiveLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorValidatingLicensekey:
            message = error.errorMessage;
            break;
        case AcuantErrorCameraUnauthorized:
            message = error.errorMessage;
            break;
        default:
            break;
    }
    [self showSimpleAlertWithMessage:message];
}
```

b Optional delegate methods

```
-(void)didPressBackButton{
    [_instance dismissCardCaptureInterface];
}

- (UIImage*)imageForBackButton{
    UIImage *image = [UIImage imageNamed:@"Back-Withe.png"];
    return image;
}

-(CGRect)frameForBackButton{
    return CGRectZero
}

- (BOOL)showBackButton{
    return YES;
}

- (BOOL)showFlashlightButton{
    return YES;
}

-(CGRect)frameForFlashlightButton{
    return CGRectZero
}

- (UIImage*)imageForFlashlightButton{
    UIImage *image = [UIImage imageNamed:@"FlashlightButton.png"];
    return image;
}

- (UIImage*)imageForHelpImageView{
    UIImage *image = [UIImage imageNamed:@"PDF417"];
    return [image imageByApplyingAlpha:0.7];
}

-(CGRect)frameForHelpImageView{
    UIImage *image = [UIImage imageNamed:@"PDF417"];
    CGRect frame = CGRectMake(self.view.frame.size.width/2 -
image.size.width/2, self.view.frame.size.height/2 -
image.size.height/3 , image.size.width, image.size.height);
    return frame;
}

-(NSString *)stringForWatermarkLabel{
    NSString *string = @"Powered by Acuant";
    return string;
}
```

```
-(CGRect)frameForWatermarkImageView{
    UIImage *image = [UIImage imageNamed:@"Logo.png"];
    CGRect frame = CGRectMake(self.view.frame.size.width/2 -
image.size.width/2, self.view.frame.size.height/2 -
image.size.height/2 + 20 , image.size.width, image.size.height);

    return frame;
}

- (NSString *)stringForBarcodeErrorMessage{
    NSString *string = @"Unable to scan the barcode?";
    return string;
}
- (NSString *)stringForBarcodeTitleError{
    NSString *string = @"Title Sample";
    return string;
}

- (int)timeForBarcodeErrorMessage{
    return 10;
}

- (BOOL)isHiddenBarcodeErrorMessage{
    return YES;
}
```

6 Processing a card

A SDK Configuration for card capture interface.

a In the header file where you'll be doing the parsing, add the following import.

```
#import <AcuantMobileSDK/AcuantMobileSDKController.h>
```

b In the same header file, implement the AcuantMobileSDKControllerProcessingDelegate.

```
@interface ISGViewController ()
<AcuantMobileSDKControllerCapturingDelegate,
AcuantMobileSDKControllerProcessingDelegate>
```

B Card processing method.

a For Driver's License Cards

In order to setup AcuantCardTypeDriverLicenseCard, set the following values.

```
- (IBAction)sendRequest:(id)sender {
    self.view.userInteractionEnabled = NO;
    [SVProgressHUD showWithStatus:@"Sending Request"];

    //Obtain the front side of the card image
    UIImage *frontSideImage = [self frontSideCardImage];
    //Obtain the back side of the card image
    UIImage *backSideImage =[self backSideCardImage];

    //Obtain the default AcuantCardProcessRequestOptions object for
the type of card you want to process (Driver's License card for this
example)
    AcuantCardProcessRequestOptions *options =
[AcuantCardProcessRequestOptions defaultRequestOptionsForCardType:
AcuantCardTypeDriversLicenseCard];

    //Optionally, configure the options to the desired value
    options.autoDetectState = YES;
    options.stateID = -1;
    options.reformatImage = YES;
    options.reformatImageColor = 0;
    options.DPI = 150.0f;
    options.cropImage = NO;
    options.faceDetection = YES;
    options.signatureDetection = YES;
    options.region = _regionID;
    options.sourceImage = 101;

    // Now, perform the request
    [_instance processFrontCardImage:frontSideImage
                      BackCardImage:backSideImage
                      andStringData:_barcodeString
                       withDelegate:self
                        withOptions:options];

}
```

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
Gray scale 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 AcuantCardTypeMedicalInsuranceCard, just set the following values.

```
- (IBAction)sendRequest:(id)sender {
    self.view.userInteractionEnabled = NO;
    [SVProgressHUD showWithStatus:@"Sending Request"];

    //Obtain the front side of the card image
    UIImage *frontSideImage = [self frontSideCardImage];
    //Optionally, Obtain the back side of the image
    UIImage *backSideImage =[self backSideCardImage];

    //Obtain the default AcuantCardProcessRequestOptions object for
the type of card you want to process (Medical Insurance card for this
```

```
example)
    AcuantCardProcessRequestOptions *options =
[AcuantCardProcessRequestOptions defaultRequestOptionsForCardType:
AcuantCardTypeMedicalInsuranceCard];

    //Optionally, configure the options to the desired value
    options.reformatImage = YES;
    options.reformatImageColor = 0;
    options.DPI = 150.0f;
    options.cropImage = NO;

    // Now, perform the request
    [_instance processFrontCardImage:frontSideImage
                        BackCardImage:backSideImage
                        andStringData:nil
                         withDelegate:self
                          withOptions:options];

}
```

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
Gray scale 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 AcuantCardTypePassportCard, just set the following values.

```
- (IBAction)sendRequest:(id)sender {
    self.view.userInteractionEnabled = NO;
    [SVProgressHUD showWithStatus:@"Sending Request"];

    //Obtain the front side of the card image
    UIImage *frontSideImage = [self frontSideCardImage];
```

```
    //Obtain the default AcuantCardProcessRequestOptions object for
the type of card you want to process (Passport card for this example)
    AcuantCardProcessRequestOptions *options =
[AcuantCardProcessRequestOptions defaultRequestOptionsForCardType:
AcuantCardTypePasssportCard];

    //Optionally, configure the options to the desired value
    options.reformatImage = YES;
    options.reformatImageColor = 0;
    options.DPI = 150.0f;
    options.cropImage = NO;
    options.faceDetection = YES;
    options.signatureDetection = YES;

    // Now, perform the request
    [_instance processFrontCardImage:frontSideImage
                       BackCardImage:nil
                       andStringData:nil
                        withDelegate:self
                         withOptions:options];

}
```

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
Gray scale 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 AcuantMobileSDKControllerProcessingDelegate protocol to handle the processing.

a For Driver's License Cards

If using the AcuantCardTypeDriversLicenseCard, add the following code:

```
#pragma mark -
#pragma mark CardProcessing Delegate
-(void)didFinishProcessingCardWithResult:(AcuantCardResult *)result{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    UIImage *faceimage;
    UIImage *signatureImage;
    UIImage *frontImage;
    UIImage *backImage;
    AcuantDriversLicenseCard *data =
(AcuantDriversLicenseCard*)result;
    message =[NSString stringWithFormat:@"First Name - %@ \nMiddle
Name - %@ \nLast Name - %@ \nName Suffix - %@ \nID - %@ \nLicense - %@
\nDOB Long - %@ \nDOB Short - %@ \nDate Of Birth Local - %@ \nIssue
Date Long - %@ \nIssue Date Short - %@ \nIssue Date Local - %@
\nExpiration Date Long - %@ \nExpiration Date Short - %@ \nEye Color -
%@ \nHair Color - %@ \nHeight - %@ \nWeight - %@ \nAddress - %@
\nAddress 2 - %@ \nAddress 3 - %@ \nAddress 4 - %@ \nAddress 5 - %@
\nAddress 6  - %@ \nCity - %@ \nZip - %@ \nState - %@ \nCounty - %@
\nCountry Short - %@ \nCountry Long - %@ \nClass - %@ \nRestriction -
%@ \nSex - %@ \nAudit - %@ \nEndorsements - %@ \nFee - %@ \nCSC - %@
\nSigNum - %@ \nText1 - %@ \nText2 - %@ \nText3 - %@ \nType - %@ \nDoc
Type - %@ \nFather Name - %@ \nMother Name - %@ \nNameFirst_NonMRZ -
%@ \nNameLast_NonMRZ - %@ \nNameLast1 - %@ \nNameLast2 - %@
\nNameMiddle_NonMRZ - %@ \nNameSuffix_NonMRZ - %@ \nNationality - %@
\nOriginal - %@ \nPlaceOfBirth - %@ \nPlaceOfIssue - %@ \nSocial
Security - %@ \nIsAddressCorrected - %@ \nIsAddressVerified - %@",
data.nameFirst, data.nameMiddle, data.nameLast, data.nameSuffix,
data.licenceId, data.license, data.dateOfBirth4, data.dateOfBirth,
data.dateOfBirthLocal, data.issueDate4, data.issueDate,
data.issueDateLocal, data.expirationDate4, data.expirationDate,
data.eyeColor, data.hairColor, data.height, data.weight, data.address,
data.address2, data.address3, data.address4, data.address5,
data.address6, data.city, data.zip, data.state, data.county,
data.countryShort, data.idCountry, data.licenceClass,
data.restriction, data.sex, data.audit, data.endorsements, data.fee,
data.CSC, data.sigNum, data.text1, data.text2, data.text3, data.type,
data.docType, data.fatherName, data.motherName, data.nameFirst_NonMRZ,
data.nameLast_NonMRZ, data.nameLast1, data.nameLast2,
data.nameMiddle_NonMRZ, data.nameSuffix_NonMRZ, data.nationality,
data.original, data.placeOfBirth, data.placeOfIssue,
data.socialSecurity, data.isAddressCorrected, data.isAddressVerified];
if (_region == AcuantCardRegionUnitedStates || _region ==
```

```
AcuantCardRegionCanada) {
            message = [NSString stringWithFormat:@"%@ \nIsBarcodeRead
- %@ \nIsIDVerified - %@ \nIsOcrRead - %@ \nDocument Verification
Confidence Rating - %@", message, data.isBarcodeRead,
data.isIDVerified, data.isOcrRead, data.documentVerificationRating];
}

    faceimage = [UIImage imageWithData:data.faceImage];
    signatureImage = [UIImage imageWithData:data.signatureImage];
    frontImage = [UIImage imageWithData:data.licenceImage];
    backImage = [UIImage imageWithData:data.licenceImageTwo];
-(void)didFailWithError:(AcuantError *)error{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    switch (error.errorType) {
        case AcuantErrorTimedOut:
            message = error.errorMessage;
            break;
        case AcuantErrorUnknown:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToProcess:
            message = error.errorMessage;
            break;
        case AcuantErrorInternalServerError:
            message = error.errorMessage;
            break;
        case AcuantErrorCouldNotReachServer:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToAuthenticate:
            message = error.errorMessage;
            break;
        case AcuantErrorAutoDetectState:
            message = error.errorMessage;
            break;
        case AcuantErrorWebResponse:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToCrop:
            message = error.errorMessage;
            break;
        case AcuantErrorInvalidLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorInactiveLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorAccountDisabled:
```

```
            message = error.errorMessage;
            break;
        case AcuantErrorOnActiveLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorValidatingLicensekey:
            message = error.errorMessage;
            break;
        case AcuantErrorCameraUnauthorized:
            message = error.errorMessage;
            break;
        default:
            break;
    }
    [self showSimpleAlertWithMessage:message];
}
```

b For Medical Insurance Cards

If using the AcuantCardTypeMedicalInsuranceCard, add the following code:

```
#pragma mark -
#pragma mark CardProcessing Delegate
-(void)didFinishProcessingCardWithResult:(AcuantCardResult *)result{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    UIImage *faceimage;
    UIImage *signatureImage;
    UIImage *frontImage;
    UIImage *backImage;
    AcuantMedicalInsuranceCard *data =
(AcuantMedicalInsuranceCard*)result;
    message =[NSString stringWithFormat:@"First Name - %@ \nLast Name
- %@ \nMiddle Name - %@ \nMemberID - %@ \nGroup No. - %@ \nContract
Code - %@ \nCopay ER - %@ \nCopay OV - %@ \nCopay SP - %@ \nCopay UC -
%@ \nCoverage - %@ \nDate of Birth - %@ \nDeductible - %@ \nEffective
Date - %@ \nEmployer - %@ \nExpire Date - %@ \nGroup Name - %@
\nIssuer Number - %@ \nOther - %@ \nPayer ID - %@ \nPlan Admin - %@
\nPlan Provider - %@ \nPlan Type - %@ \nRX Bin - %@ \nRX Group - %@
\nRX ID - %@ \nRX PCN - %@ \nTelephone - %@ \nWeb - %@ \nEmail - %@
\nAddress - %@ \nCity - %@ \nZip - %@ \nState - %@", data.firstName,
data.lastName, data.middleName, data.memberId, data.groupNumber,
data.contractCode, data.copayEr, data.copayOv, data.copaySp,
data.copayUc, data.coverage, data.dateOfBirth, data.deductible,
data.effectiveDate, data.employer, data.expirationDate,
data.groupName, data.issuerNumber, data.other, data.payerId,
data.planAdmin, data.planProvider, data.planType, data.rxBin,
data.rxGroup, data.rxId, data.rxPcn, data.phoneNumber,
```

```
data.webAddress, data.email, data.fullAddress, data.city, data.zip,
data.state];

    frontImage = [UIImage imageWithData:data.reformattedImage];
    backImage = [UIImage imageWithData:data.reformattedImageTwo];


-(void)didFailWithError:(AcuantError *)error{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    switch (error.errorType) {
        case AcuantErrorTimedOut:
            message = error.errorMessage;
            break;
        case AcuantErrorUnknown:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToProcess:
            message = error.errorMessage;
            break;
        case AcuantErrorInternalServerError:
            message = error.errorMessage;
            break;
        case AcuantErrorCouldNotReachServer:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToAuthenticate:
            message = error.errorMessage;
            break;
        case AcuantErrorAutoDetectState:
            message = error.errorMessage;
            break;
        case AcuantErrorWebResponse:
            message = error.errorMessage;
            break;
        case AcuantErrorUnableToCrop:
            message = error.errorMessage;
            break;
        case AcuantErrorInvalidLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorInactiveLicenseKey:
            message = error.errorMessage;
            break;
        case AcuantErrorAccountDisabled:
            message = error.errorMessage;
            break;
        case AcuantErrorOnActiveLicenseKey:
            message = error.errorMessage;
```

```
            break;
        case AcuantErrorValidatingLicensekey:
            message = error.errorMessage;
            break;
        case AcuantErrorCameraUnauthorized:
            message = error.errorMessage;
            break;

        default:
            break;
    }
    [self showSimpleAlertWithMessage:message];
}
```

c For Passport.

If using the AcuantCardTypePassportCard, add the following code:

```
#pragma mark -
#pragma mark CardProcessing Delegate
-(void)didFinishProcessingCardWithResult:(AcuantCardResult *)result{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    UIImage *faceimage;
    UIImage *signatureImage;
    UIImage *frontImage;
    UIImage *backImage;
    AcuantPassaportCard *data = (AcuantPassaportCard*)result;
    message =[NSString stringWithFormat:@"First Name - %@ \nMiddle
Name - %@ \nLast Name - %@ \nPassport Number - %@ \nPersonal Number -
%@ \nSex - %@ \nCountry Long - %@ \nNationality Long - %@ \nDOB Long -
%@ \nIssue Date Long - %@ \nExpiration Date Long - %@ \nPlace of Birth
- %@", data.nameFirst, data.nameMiddle, data.nameLast,
data.passportNumber, data.personalNumber, data.sex, data.countryLong,
data.nationalityLong, data.dateOfBirth4, data.issueDate4,
data.expirationDate4, data.end_POB];

    faceimage = [UIImage imageWithData:data.faceImage];
    frontImage = [UIImage imageWithData:data.passportImage];


-(void)didFailWithError:(AcuantError *)error{
    self.view.userInteractionEnabled = YES;
    [SVProgressHUD dismiss];
    NSString *message;
    switch (error.errorType) {
        case AcuantErrorTimedOut:
            message = error.errorMessage;
            break;
```

```
            case AcuantErrorUnknown:
                message = error.errorMessage;
                break;
            case AcuantErrorUnableToProcess:
                message = error.errorMessage;
                break;
            case AcuantErrorInternalServerError:
                message = error.errorMessage;
                break;
            case AcuantErrorCouldNotReachServer:
                message = error.errorMessage;
                break;
            case AcuantErrorUnableToAuthenticate:
                message = error.errorMessage;
                break;
            case AcuantErrorAutoDetectState:
                message = error.errorMessage;
                break;
            case AcuantErrorWebResponse:
                message = error.errorMessage;
                break;
            case AcuantErrorUnableToCrop:
                message = error.errorMessage;
                break;
            case AcuantErrorInvalidLicenseKey:
                message = error.errorMessage;
                break;
            case AcuantErrorInactiveLicenseKey:
                message = error.errorMessage;
                break;
            case AcuantErrorAccountDisabled:
                message = error.errorMessage;
                break;
            case AcuantErrorOnActiveLicenseKey:
                message = error.errorMessage;
                break;
            case AcuantErrorValidatingLicensekey:
                message = error.errorMessage;
                break;
            case AcuantErrorCameraUnauthorized:
                message = error.errorMessage;
                break;
            default:
                break;
        }
        [self showSimpleAlertWithMessage:message];
    }
```

7 Change Log

If you are updating the CSSN iOS MobileSDK from version 4.4, please be careful with the names of the methods.

a Renamed methods.

The following methods were renamed.

```
/**
 Use this method to obtain an instance of the
AcuantMobileSDKController if License key is correct and show the
camera interface after the key was validated and approved
 @param key your License Key
 @param viewController the UIViewController object from which we'll
present the card capture interface
 @param delegate the delegate of the card capture interface
 @param typeCard the type of the card capture interface
 @param isBarcodeSide the side of the card and type of capture
interface
 @discussion never try to alloc/init this class, always obtain an
instance through this method.
 @return the AcuantMobileSDKController instance
 */
+
(AcuantMobileSDKController*)initAcuantMobileSDKWithLicenseKey:(NSStrin
g*)key
AndShowCardCaptureInterfaceInViewController:(UIViewController*)vc
delegate:(id<AcuantMobileSDKControllerCapturingDelegate,
AcuantMobileSDKControllerProcessingDelegate>)delegate
typeCard:(AcuantCardType)typeCard region:(AcuantCardRegion)region
isBarcodeSide:_isBarcodeSide;

/**
 Use this method to present the card capture interface.
 @param viewController the UIViewController object from which we'll
present the card capture interface
 @param delegate the delegate of the card capture interface
 @param typeCard the type of the card capture interface
 @param region the region of the card and type of capture interface
 @discussion a valid viewController is required
 */
- (void)showCameraInterfaceInViewController:(UIViewController*)vc
delegate:(id<AcuantMobileSDKControllerCapturingDelegate>)delegate
cardType:(AcuantCardType)cardType region:(AcuantCardRegion)region
isBarcodeSide:(BOOL)isBarcodeSide;
```

b New methods.

The following methods are news.

```
/**
 Called to obtain the flashlight button image displayed in the card
capture interface
 @return the flashlight button image
 @discussion if this method is not implemented or nil is returned,
we'll display a white rounded button with "flash" text
 @discussion this delegate method is only called when presenting the
card capture interface full screen. If card capture interface is
presented in a UIPopOverController, this method is not called at all
because a Cancel UIBarButtonItem in the UINavigationBar is used
instead.
 */
- (UIImage*)imageForFlashlightButton;

/**
 Called to obtain the flashlight button position in the screen.
 @return the point where the flashlight button should be positioned
 @discussion in case this method is not implemented by the delegate,
we'll set a default location for the button though we encourage you to
set the position manually.
 @discussion if your application supports multiple screen sizes then
you are in charge of returning the correct position for each screen
size.
 */
- (CGRect)frameForFlashlightButton;

/**
 Called to show or not show the flashlight button in the card capture
interface
 @return show or not show the flashlight button
 @discussion if this method is not implemented or nil is returned,
we'll display a the button with "flash" text
 */
- (BOOL)showFlashlightButton;
```