Mobile VIN Scanner SDK Overview

VINScanner SDK allows to add a quick and precise recognition of VIN barcodes in applications for iOS. The codes are successfully recognized even in case of blurriness and noise, which are often when using the camera in difficult conditions.

SDK carries out all necessary actions with the device's camera itself. SDK also performs verifications of scanned VIN codes and uses optimal amount of pixels to provide quick and reliable recognition.

Supported iOS version
SDK can be used on devices under iOS 5.0 and younger.

Supported architectures
The following architectures are supported: ARM v7, ARM v7s, ARM 64, i386, x86_64.

Supported devices
The following devices are supported:
- iPhone 3GS+;
- iPad 3+ (iPad 2 is supported, but scanning quality is very low due to bad hardware camera);
- iPad Mini;
- iPod Touch 3rd+ generation.

Trial mode
Trial (demo) mode is supported by SDK. When there is no license key added to project, SDK will function in trial mode. Trial mode is fully-functional and works the same as is production mode. Demo version is intended for developers to verify that VIN Scanner SDK works well in theirs apps.

There are two restrictions in demo mode:
1. Our logo (bees4honey) will appear on scan screen.
2. Amount of scanned VIN codes per app install is limited to 25.

Both of these restrictions will be automatically removed when you add a license key to your project.

Please see example project for demonstration of SDK usage
https://github.com/bees4honey/mobile_vin_scanner/tree/master/ios_scanner_sdk

SDK Structure

This section describes set of files you need for library usage.

ScannerController.h

Declaration of classes, protocols and constants.

scannerlicense.plist

License file. Optional. This file is generated by us upon your request. License file should be added to project files. You should store this file in the same directory with libB4HVINScanner.a static library. Without this file only trial mode is available.

libB4HVINScanner.a

VINScanner SDK static library.

Adding SDK into the project

The file with the static library and header file should be included into the project.

SDK involves dependences from the following frameworks:

- UIKit.framework
- Foundation.framework
- QuartzCore.framework
- CoreVideo.framework
- CoreMedia.framework
- AVFoundation.framework
- Security.framework

Important!

The application should be built with -ObjC key. SDK classes can't be used with iOS versions older than 5.0

SDK Usage

This section describes usage of scanning library and possibilities of scanner.

Description of classes and protocols

B4HScannerController

The main scanner class. It displays an image from the camera without any control elements. There is a bees4honey logo in a trial mode. Both landscape and portrait orientations are supported. Start of a cam and the process of recognition happens at controller's viewWillAppear. A stop and cam disability takes places at viewDidDisappear.

It is possible to check when SDK is ready for work by calling CheckReadyStatus - returns ScannerController status (readiness or error). A verification of license file, number of uses in the trial mode, presence of a cam on a device and availability of frameworks

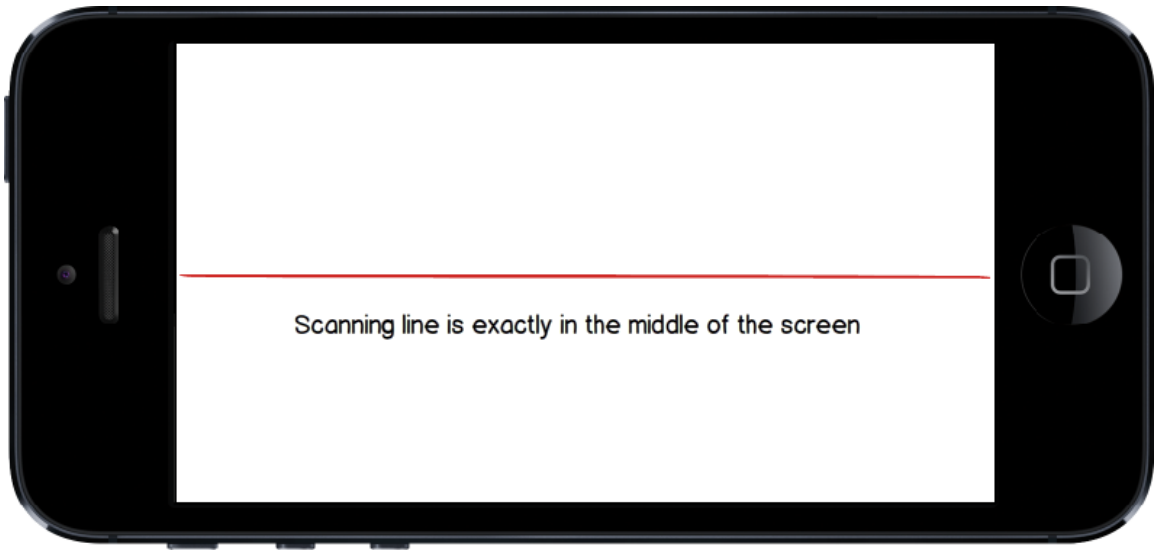
necessary for use are carried out. Check of the cam presence is not carried out on a simulator.

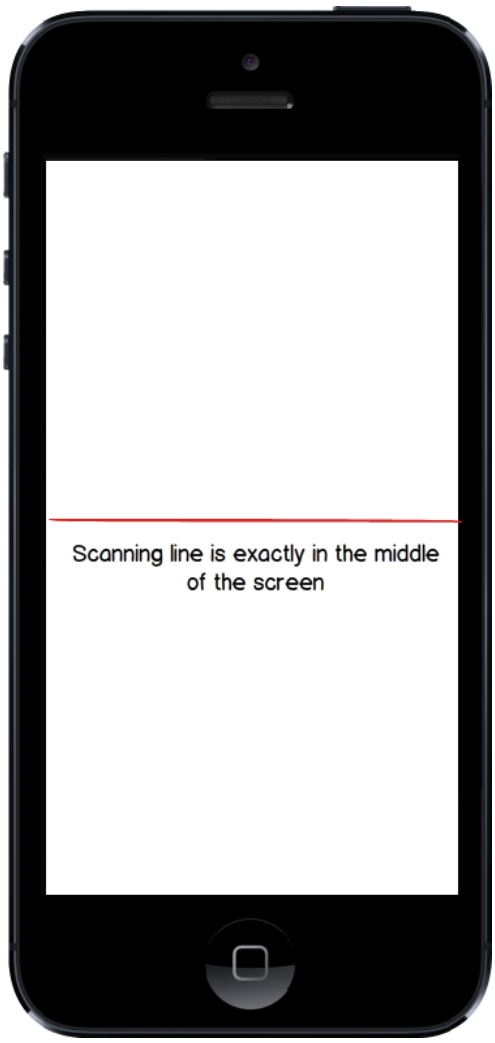
```
typedef enum
{
     B4HOrientationHorizontal = 0,
     B4HOrientationVertical = 1
}
B4HScannerOrientation;

@interface B4HScannerController : UIViewController
- (id) initWithOrientation:(B4HScannerOrientation) ScanOrientation;
- (void) startScanningWithOrientation:(B4HScannerOrientation) orientation;
- (BOOL) isRunning;
- (void) startScanning;
- (void) stopScanning;
- (B4HScannerLibraryStatus) CheckReadyStatus;

@property (nonatomic, retain) B4HCameraOverlayViewController *overlay;
@property (nonatomic, assign) id <B4HScannerDelegate> delegate;
@property (nonatomic, assign) B4HScannerOrientation scanOrientation;

@end
```

It’s possible to check whether scanning and recognition process is working, start and stop scanner by calling these functions:

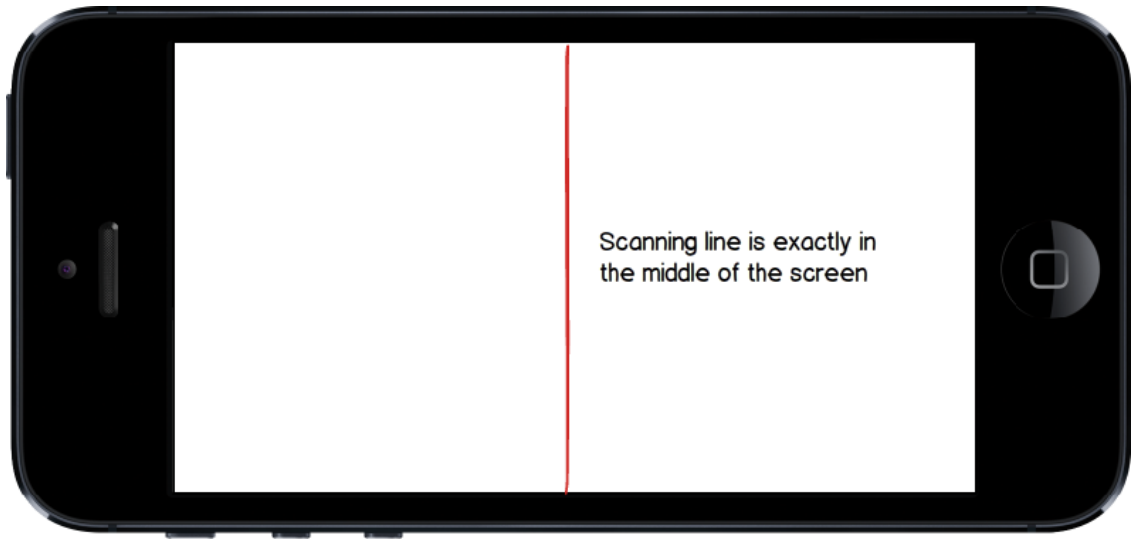
```
- (BOOL) isRunning;
- (void) startScanning;
- (void) startScanningWithOrientation:(B4HScannerOrientation) orientation;
- (void) stopScanning;
```

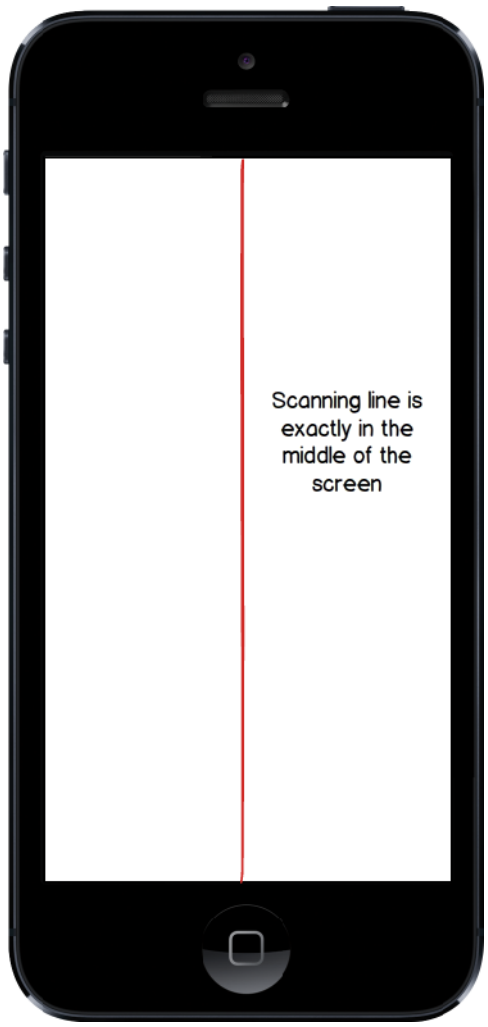
By default scanning is carried out horizontally (from left to right) along the line passing exactly in the middle of the screen in both portrait and landscape orientations.





VIN barcodes can also be scanned vertically. In some cases it may be useful, for example, when VIN barcode is located in inconvenient place and user may want to switch to vertical scanning. In this case scanning will be held along the vertical line in the middle of the screen.





You can change scanning direction to vertical either by calling startScanningWithOrientation or by setting scanOrientation property in B4HScannerController:

```
- (void) changeScanOrientation
{
    //parentScanner is instance of B4HScannerController
    if (self.parentScanner.scanOrientation == B4HOrientationVertical)
    {
        self.parentScanner.scanOrientation = B4HOrientationHorizontal;
    }
    else
    {
        self.parentScanner.scanOrientation = B4HOrientationVertical;
    }
}
```

On a simulator scanning of a code is emulated by clicking "return code" button, located in the center of the screen.

CameraOverlayViewController
The class serves for modification of scanner interface. Having inherited an arbitrary class from CameraOverlayViewController and having connected to scanner controller as an overlay property, it is possible to put view over scanner's view. Correspondingly, presentation of CameraOverlayViewController should be transparent for a display of a picture from the cam.

The use of classes inherited from CameraOverlayViewController in the application is not a necessary thing, as on default ScannerController contains CameraOverlayViewController. At the start on a simulator this controller displays a button in the centre, which imitates code scanning. ScannerController is built into the hierarchy of app controllers, but not CameraOverlayViewController.

```
@interface CameraOverlayViewController : UIViewController { }
@property (nonatomic, assign) ScannerController *parentScanner;
@end
```

ScannerDelegate
The protocol of a delegate which receives the code recognized by the scanner. Any controller, not only CameraOverlayViewController, can be in the role of a delegate.

```
@protocol ScannerDelegate <NSObject>
@required
- (void)scanner:(ScannerController *)scanner gotCode:(NSString *)code;
@end
```