iOS SDK

Anubhav Saggi, Rajan Bala
hiku labs, inc.

Revision History

Date	Author	Description	Version
10/15/2014	Rajan Bala	Initial Draft of the SDK	1.0
11/05/2014	Anubhav Saggi	Changed the following: • An initial screen that allows a user to input his email address to connect his Hiku. • An additional delegate method that notifies on a successful third-party application authorization by Hiku. • An API to return the application token, if available. • An API to explicitly logout a user if he is logged in. • Removed the sample code; reference the code in the sample setup app instead.	2.0
11/11/2014	Anubhav Saggi	Added the following: • UI/UX enhancements. • An additional API to view the “Tips” flow.	2.1
12/29/2014	Anubhav Saggi	Added the following: • New login, device setup, and onboarding flows. • Conversion of the setup application from an iPhone-only to a Universal binary (for iPad support).	3.0
02/01/2015	Anubhav Saggi	Added the following: • New APIs to switch the presentation style of UX flows from modal to push (and vice-versa).	3.1
03/12/2015	Anubhav Saggi	Added the following: • New delegate method that’s fired when the user completes the tips/onboarding flow.	3.1

01/25/2016	Anubhav Saggi	Added the following: • Sections describing client integration steps and known issues. • New delegate method that's fired when a user logs out. • New API that creates a new user account or logs an existing user in.	3.2

Table Of Contents

List of Figures

hiku iOS SDK

1.0 Background

The hiku iOS SDK enables 3rd party applications to incorporate the hiku device setup and onboarding experiences into their own mobile applications. Connection is established via the "BlinkUp" process: a flash emanates from the smartphone screen to communicate with the hiku device and thereby pair it with a local Wi-Fi network. In the past, only hiku's native iOS application could setup a hiku device and connect it to a Wi-Fi network.

2.0 Prerequisites

2.1 Supported iOS Versions

iOS versions 7.0+ are currently supported by the SDK **however this is subject to change in the future to iOS 8.2+ (mainly to accommodate watchOS 2.0+ support in the future).**

2.2 Libraries

The following is a modified snapshot of the `Podfile.lock` file that shows the external libraries required by the SDK. Most have already been included with the `HikuSetupApp` setup application and missing ones can either be added manually or installed via CocoaPods:

```
- AFNetworking (2.6.3):
  - AFNetworking/NSURLConnection (= 2.6.3)
  - AFNetworking/NSURLSession (= 2.6.3)
  - AFNetworking/Reachability (= 2.6.3)
  - AFNetworking/Security (= 2.6.3)
  - AFNetworking/Serialization (= 2.6.3)
  - AFNetworking/UIKit (= 2.6.3)
- AFNetworking/NSURLConnection (2.6.3):
  - AFNetworking/Reachability
  - AFNetworking/Security
  - AFNetworking/Serialization
- AFNetworking/NSURLSession (2.6.3):
  - AFNetworking/Reachability
  - AFNetworking/Security
  - AFNetworking/Serialization
- AFNetworking/Reachability (2.6.3)
- AFNetworking/Security (2.6.3)
- AFNetworking/Serialization (2.6.3)
- AFNetworking/UIKit (2.6.3):
  - AFNetworking/NSURLConnection
  - AFNetworking/NSURLSession
- libPusher (1.6.1):
  - libPusher/Core (= 1.6.1)
```

```
- libPusher/Core (1.6.1):
  - SocketRocket (= 0.4.1)
- Mixpanel (2.9.1):
  - Mixpanel/Mixpanel (= 2.9.1)
- Mixpanel/Mixpanel (2.9.1)
- NSData+Base64 (1.0.0)
- nv-ios-http-status (0.0.1)
- SocketRocket (0.4.1)
- TTTAttributedLabel (1.13.4)
- UICKeyChainStore (2.1.0)
```

3.0 SDK Core Functionality

The current release of the SDK has the bare minimum support to ensure that a hiku device can be setup from within a 3rd party mobile application. The SDK is a closed source library that is linked with the third-party mobile application. hiku owns the user experience (UX) of the device setup and would customize the experience where deemed necessary. The SDK requires 3rd parties to use the application ID and shared secret provided to them by hiku labs, inc. to authenticate and start the setup process.

3.1 Account Creation and Login User Experience

Users must have an account in the hiku cloud in order to access hiku services. Via the SDK, 3rd parties provide a valid email address to identify the user uniquely during setup. The SDK will attempt to create an account in the hiku cloud in the background to start the setup process. Upon successful account creation, the SDK will take the user to the device setup screen where the user can select the Wireless Fidelity (Wi-Fi) credentials to connect the hiku device to a local Wi-Fi network. In the event that an account already exists with the provided email address, the user will be presented with the login screen shown in Figure 2. In order to continue with the setup the user must enter the password associated with the account or request a new password.

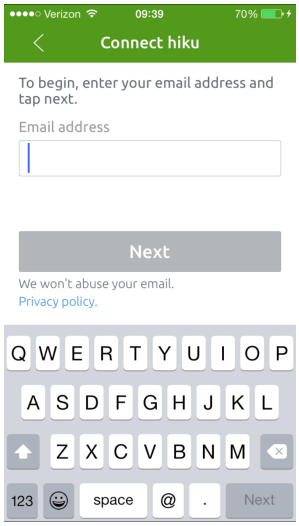


Figure 1 Email prompt to initiate the sign in flow

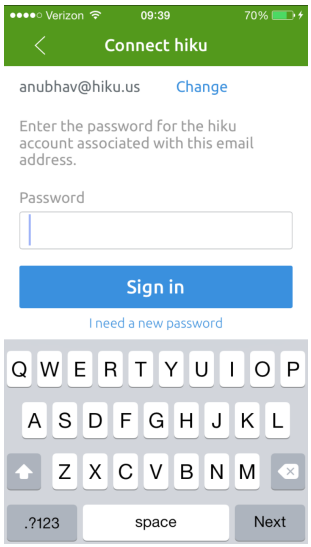


Figure 2 Login prompt after detecting that the user already exists

To directly create an account for a new user or log an existing user in, the `loginUserWithEmail` API can be called as follows:

```
- (void)loginUserWithEmail:(NSString *)email password:(NSString *)password;
```

If no login token exists locally for a user and an empty password is supplied, a new account with the specified email will be created. In all other cases, the existing user will be logged in with the specified email and password combination. The `userAuthenticationStatus` and `applicationAuthorizationStatus` delegate callbacks will be subsequently called to indicate whether or not the account creation or login completed successfully and whether or not an application token was successfully generated, respectively.

3.2 Device Setup User Experience

Once the user has successfully authenticated either via a successful first-time account creation or via logging in, the user will be presented with the device setup UX flow as delineated in the figures below.

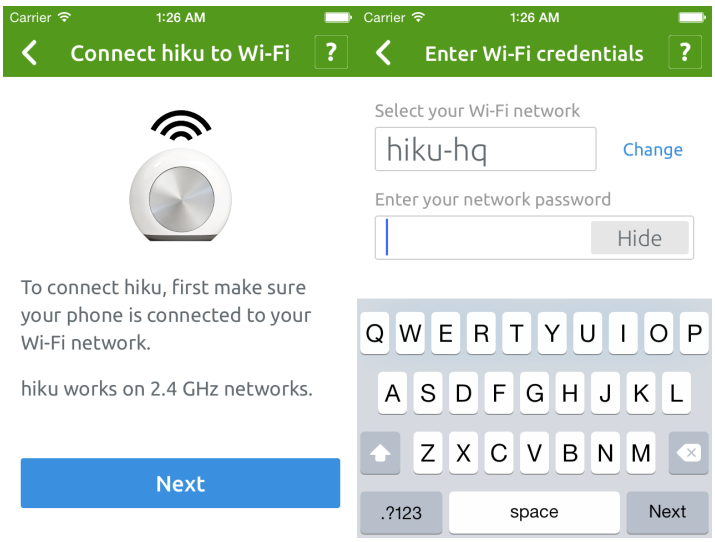


Figure 3 Flow to enter Wi-Fi credentials

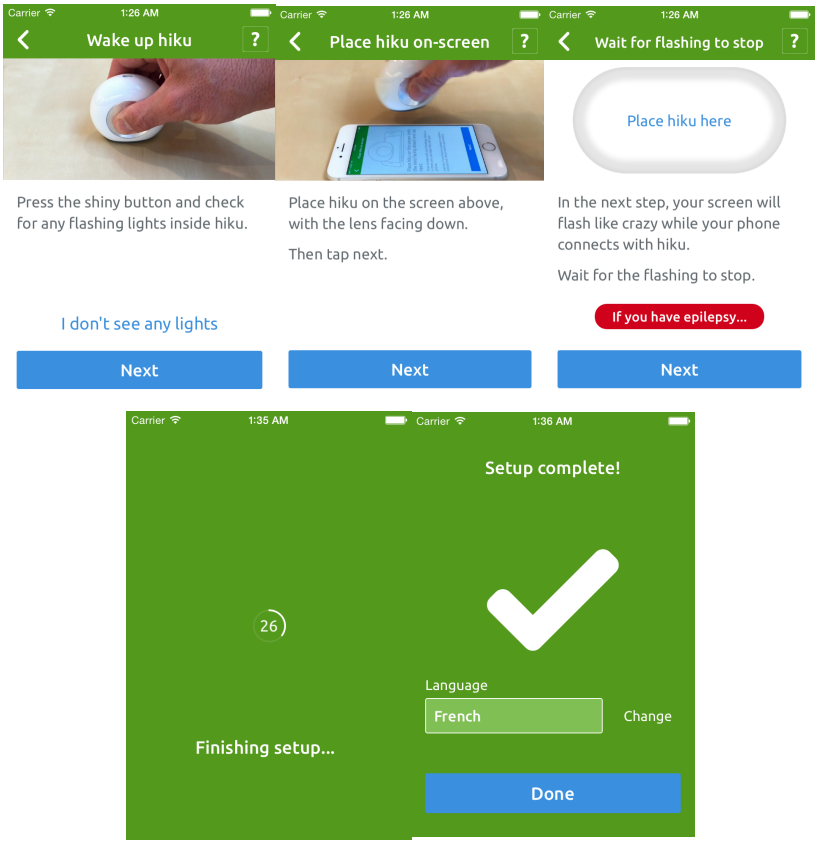


Figure 4 Flow for device BlinkUp

3.3 Tips/Onboarding User Experience

Upon a successful device BlinkUp, the user will be taken to the onboarding UX to familiarize himself with how to use a hiku. The following screenshots illustrate the current onboarding

experience presented to the user. Exiting this UX signals that the user successfully completed device setup.

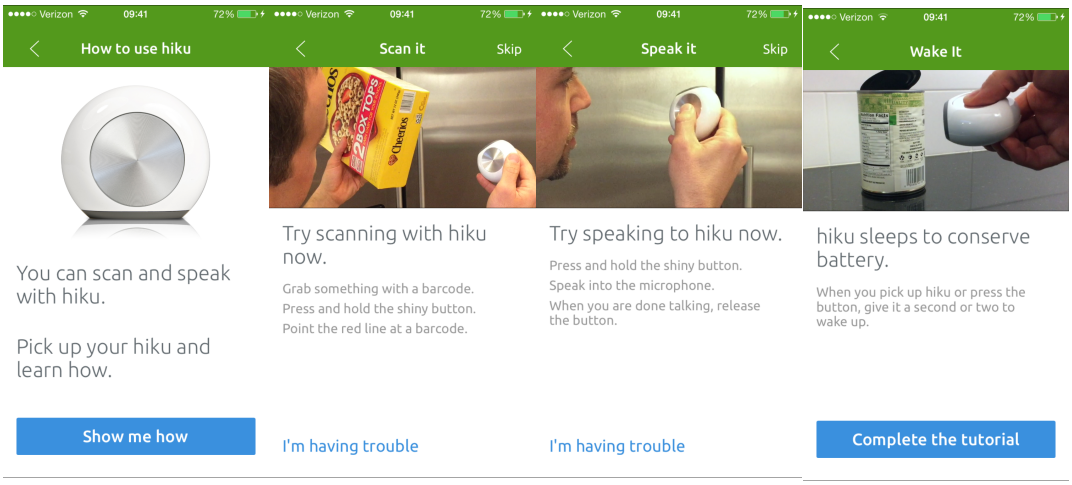


Figure 5 Optionally-presented onboarding flow

The onboarding flow can also be invoked separately via the following API (that wraps an identically named API in the SDK):

```
- (void)launchTipsFlow;
```

3.4 Explicitly Logging a User Out

To explicitly log a user out, the following API should be called (that wraps an identically named API in the SDK):

```
- (void)logoutUser;
```

3.5 Support

The SDK provides support in the form of various embedded email composition screens as well as password recovery flows for existing users who have forgotten their usernames and/or passwords. In order to use embedded email support, the smartphone must already have a valid email account setup. The password recovery flows, on the other hand, take the user to a mobile website that can help them recover their account credentials.

4.0 Events and Notifications

The following are the kinds of callback functions implemented in the SDK. These different kinds of events and notifications are delivered through delegate methods defined in the `HKSetupDelegate` protocol specified in the `HKSetupSDK.h` header file.

4.1 User Authentication Status Callback

A user's authentication status will be provided to the delegate application via the `userAuthenticationStatus` callback function indicating whether the user was successfully authenticated by the hiku cloud or not:

```
- (void)userAuthenticationStatus:(BOOL)success sdk:(HKSetupSDK
*)sdk;
```

4.2 Device Setup Status Callback

A device's setup status will be provided to the delegate application via the `deviceSetupStatus` callback function indicating whether or not a device was successfully added. A `NO` or `FALSE` in the success flag indicates that the device setup didn't complete due to incorrect Wi-Fi credentials or a loss of network connectivity:

```
- (void)deviceSetupStatus:(BOOL)success sdk:(HKSetupSDK *)sdk;
```

4.3 SDK Exit Status Callback

Upon successfully setting up a device, the SDK will exit out to the calling application. However, if the user cancels the setup process for any reason (e.g. closing the device setup flow), the `userCancelledSetup` function will be called to indicate so:

```
- (void)userCancelledSetup:(HKSetupSDK *)sdk;
```

4.4 Application Authorization Status Callback and Token Retrieval

A status denoting whether or not a third-party application is authorized by hiku will be provided to the delegate application via the `applicationAuthorizationStatus` callback function signature as follows:

```
- (void)applicationAuthorizationStatus:(BOOL)success
sdk:(HKSetupSDK *)sdk;
```

If the success status flag holds a `YES` or `TRUE` value, the `sdk` object can be further queried for the application token via the following API signature:

```
- (NSString *)getApplicationTokenForUser;
```

4.5 Tips/Onboarding Completion Callback

Upon successfully completing the tips/onboarding flow, the `userCompletedTutorial` function will be called to indicate so:

```
- (void)userCompletedTutorial:(HKSetupSDK *)sdk;
```

4.6 User Logout Callback

When a user logs out, the `userLoggedOut` function will be called to indicate so:

```
- (void)userLoggedOut:(HKSetupSDK *)sdk;
```

5.0 Extra Functionality

5.1 Supported Presentation Styles

The SDK currently supports presenting view controllers modally and pushing onto an existing navigation controller. The default presentation style is modal. The following APIs allow changing the device and onboarding UX to support a specific presentation style, respectively:

```
- (void)startSetup:(UIViewController *)withViewController
withPresentationStyle:(SDKHKPresentationStyle)presentationStyle;

- (void)launchTipsFlow:(UIViewController *)withViewController
withPresentationStyle:(SDKHKPresentationStyle)presentationStyle;
```

5.2 Landing Screen with Logo

The SDK allows displaying third-party logos as the first screen of the device setup UX. This can be specified by explicitly setting the `@property (strong, nonatomic) UIImage* partner_logo` property.

5.3 Toggling the Status Bar

The SDK allows enabling or disabling the status bar across all screens. This can be specified by explicitly setting the `@property (nonatomic, setter=setShowStatusBar:) BOOL show_status_bar` property.

6.0 Sample Code

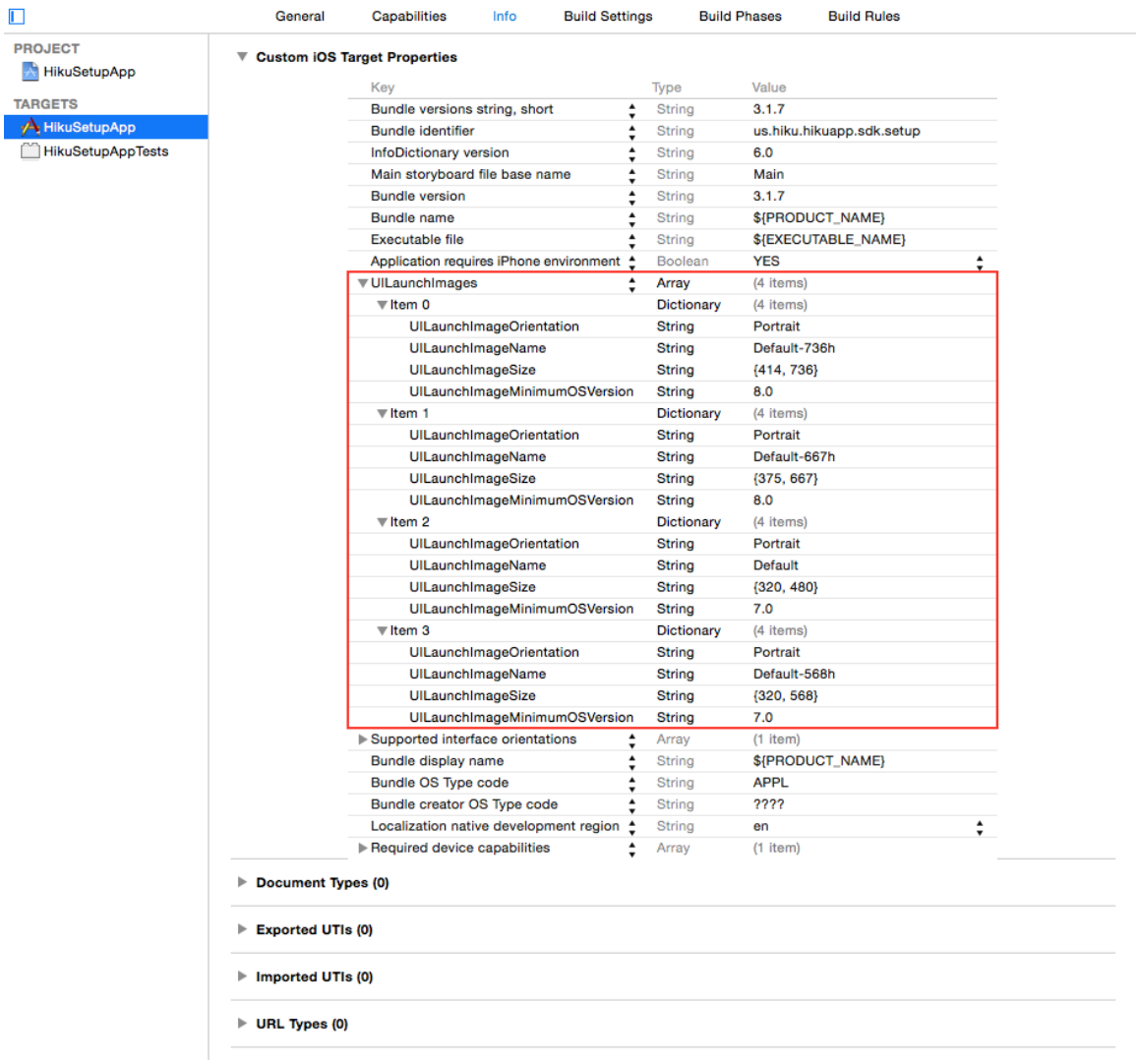
A demo app named `HikuSetupApp` has been provided as a proof-of-concept and showcases how best to integrate with the SDK. The included `HKSetupSDKFramework.framework` has everything necessary to integrate with hiku, including a resource bundle with all graphics, custom fonts, and audio tones.

7.0 Client Integration

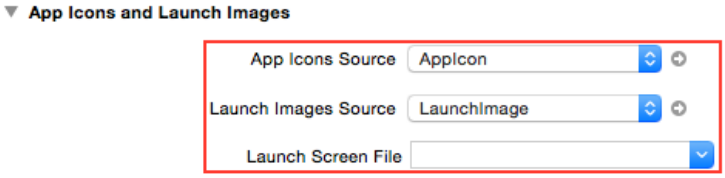
The following changes are necessary in the client application to get the best experience out of the hiku SDK:

7.1 Preventing Scaling Mode

To prevent scaling mode from being applied by iOS, the `UILaunchImages` key should be specified in the “Info” tab of the main application target with the following values:



Alternatively, scaling mode can be turned off by specifying custom launch images in the following section of the “General” tab of the main application target:



8.0 Known Issues

The following are known issues in the hiku SDK:

8.1 In-call Status Bar Issues

Currently when a modal screen is presented over a view controller embedded in a navigation controller and the double-height, in-call status bar is activated in the modal screen, dismissing the modal screen causes the in-call status bar to erroneously show on top of the navigation bar.

Conversely, when the in-call status bar is activated on the view controller embedded in the navigation controller and then deactivated in the presented modal screen, dismissing the modal screen causes a black status bar to show above the navigation bar.