

hiku

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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • UI/UX enhancements. • An additional API to view the "Tips" flow. 	2.1
12/29/2014	Anubhav Saggi	Added the following: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • New delegate method that's fired when the user completes the tips/onboarding flow. 	3.1
04/08/2015	Anubhav Saggi	Added the following: <ul style="list-style-type: none"> • Sections describing client integration steps and known issues. 	3.2

Table Of Contents

1.0	Background.....	5
2.0	Prerequisites.....	5
2.1	Supported iOS Versions.....	5
2.2	Libraries	5
3.0	SDK Core Functionality	5
3.1	Account Creation and Login User Experience	6
3.2	Device Setup User Experience	7
3.3	Tips/Onboarding User Experience	8
3.4	Explicitly Logging a User Out	8
3.5	Support.....	8
4.0	Events and Notifications.....	9
4.1	User Authentication Status Callback	9
4.2	Device Setup Status Callback	9
4.3	SDK Exit Status Callback	9
4.4	Application Authorization Status Callback and Token Retrieval.....	9
4.5	Tips/Onboarding Completion Callback.....	10
5.0	Extra Functionality.....	10
5.1	Supported Presentation Styles	10
5.2	Toggling the Status Bar	10
6.0	Sample Code.....	10
7.0	Client Integration.....	11
7.1	Preventing Scaling Mode	11
8.0	Known Issues	12
8.1	In-call Status Bar Issues	12

List of Figures

Figure 1 Email prompt to initiate the sign in flow	6
Figure 2 Login prompt after detecting that the user already exists.....	6
Figure 3 Flow to enter Wi-Fi credentials.....	7
Figure 4 Flow for device BlinkUp	7
Figure 5 Optionally-presented onboarding flow	8

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.

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:

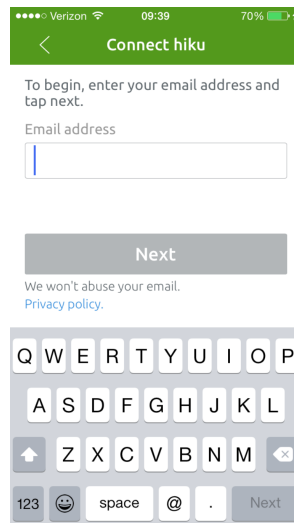
- AFNetworking (1.3.4)
- FlurrySDK (4.4.0):
 - FlurrySDK/FlurrySDK (= 4.4.0)
- FlurrySDK/FlurrySDK (4.4.0)
- libPusher (1.6-alpha):
 - libPusher/Core (= 1.6-alpha)
- libPusher/Core (1.6-alpha):
 - SocketRocket (= 0.3.1-beta2)
- Mixpanel (2.6.2):
 - Mixpanel/MPCategoryHelpers (= 2.6.2)
- Mixpanel/MPCategoryHelpers (2.6.2)
- NSData+Base64 (1.0.0)
- nv-ios-http-status (0.0.1)
- TTTAttributedLabel (1.10.0)
- UICKeyChainStore (2.0.4)

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.

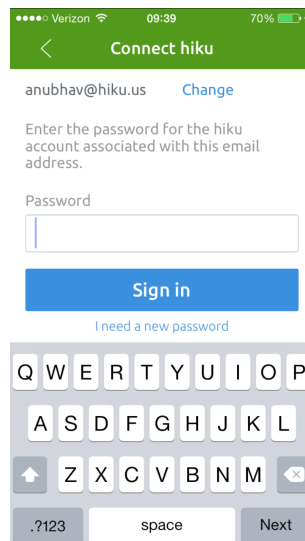
3.1 Account Creation and Login User Experience

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.



The screenshot shows a mobile app interface titled "Connect hiku". At the top, there's a status bar with "Verizon", signal strength, time "09:39", and battery "70%". Below the title bar, there's a back arrow and the title "Connect hiku". The main text says "To begin, enter your email address and tap next." Below this is a label "Email address" and an empty text input field. Under the input field is a grey "Next" button. Below the button, it says "We won't abuse your email." followed by a blue link "Privacy policy." At the bottom is a QWERTY keyboard with a "Next" button on the right.

Figure 1 Email prompt to initiate the sign in flow



The screenshot shows the same "Connect hiku" screen, but now the email "anubhav@hiku.us" is entered in the input field, and there's a blue "Change" link next to it. The main text says "Enter the password for the hiku account associated with this email address." Below this is a label "Password" and an empty text input field. Under the input field is a blue "Sign in" button. Below the button, it says "I need a new password" in blue. At the bottom is a QWERTY keyboard with a "Next" button on the right.

Figure 2 Login prompt after detecting that the user already exists

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.

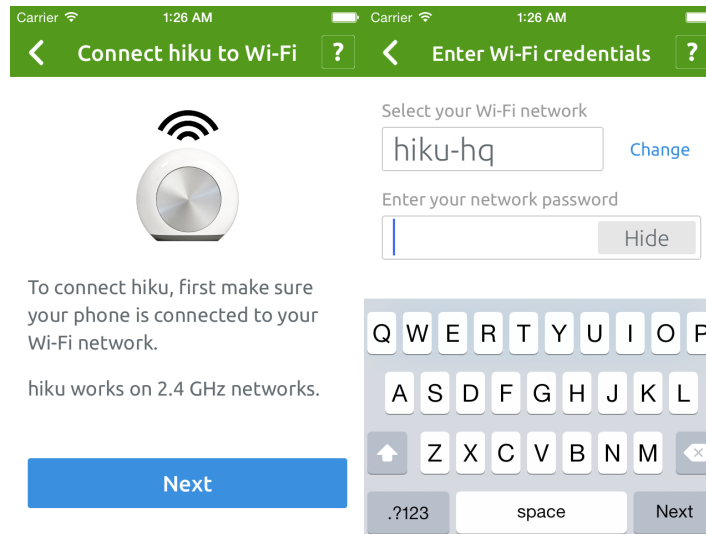


Figure 3 Flow to enter Wi-Fi credentials

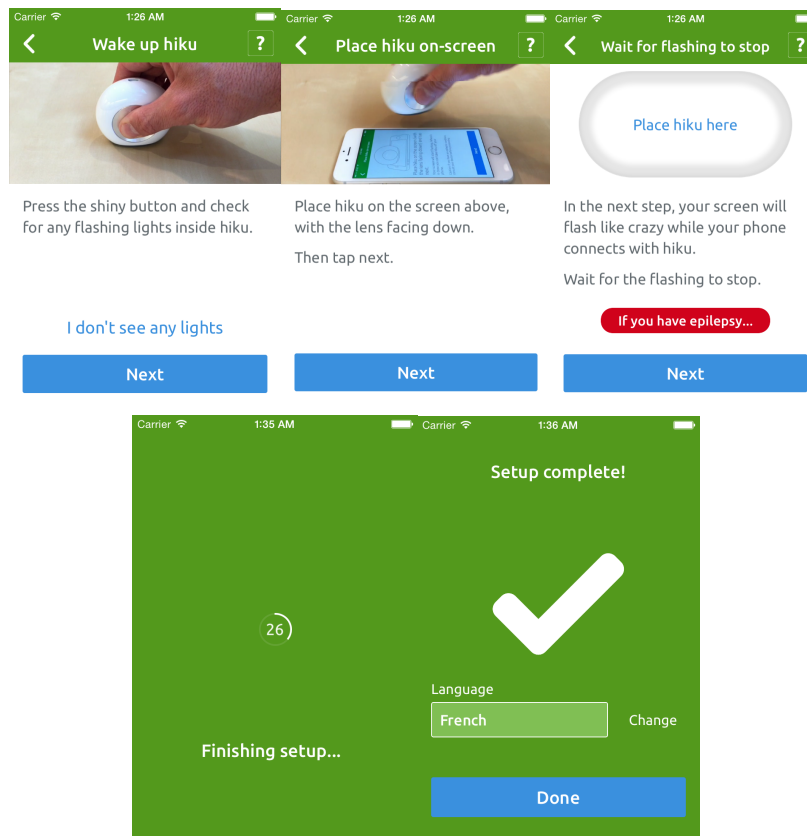


Figure 4 Flow for device BlinkUp

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.

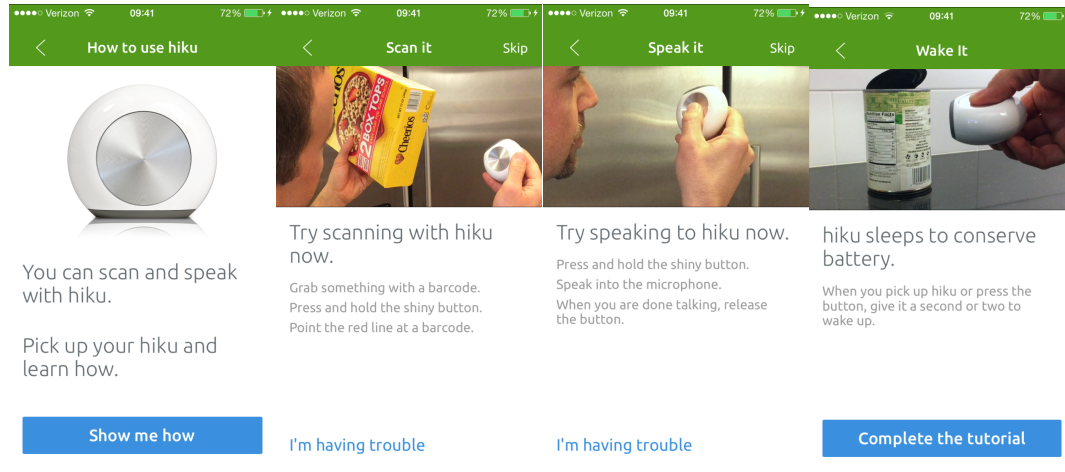


Figure 5 Optionally-presented onboarding flow

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;`

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 *)viewController
withPresentationStyle:(SDKHKPresentationStyle)presentationStyle;`

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

5.2 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=showStatusBar:) BOOL show_status_bar` property.

6.0 Sample Code

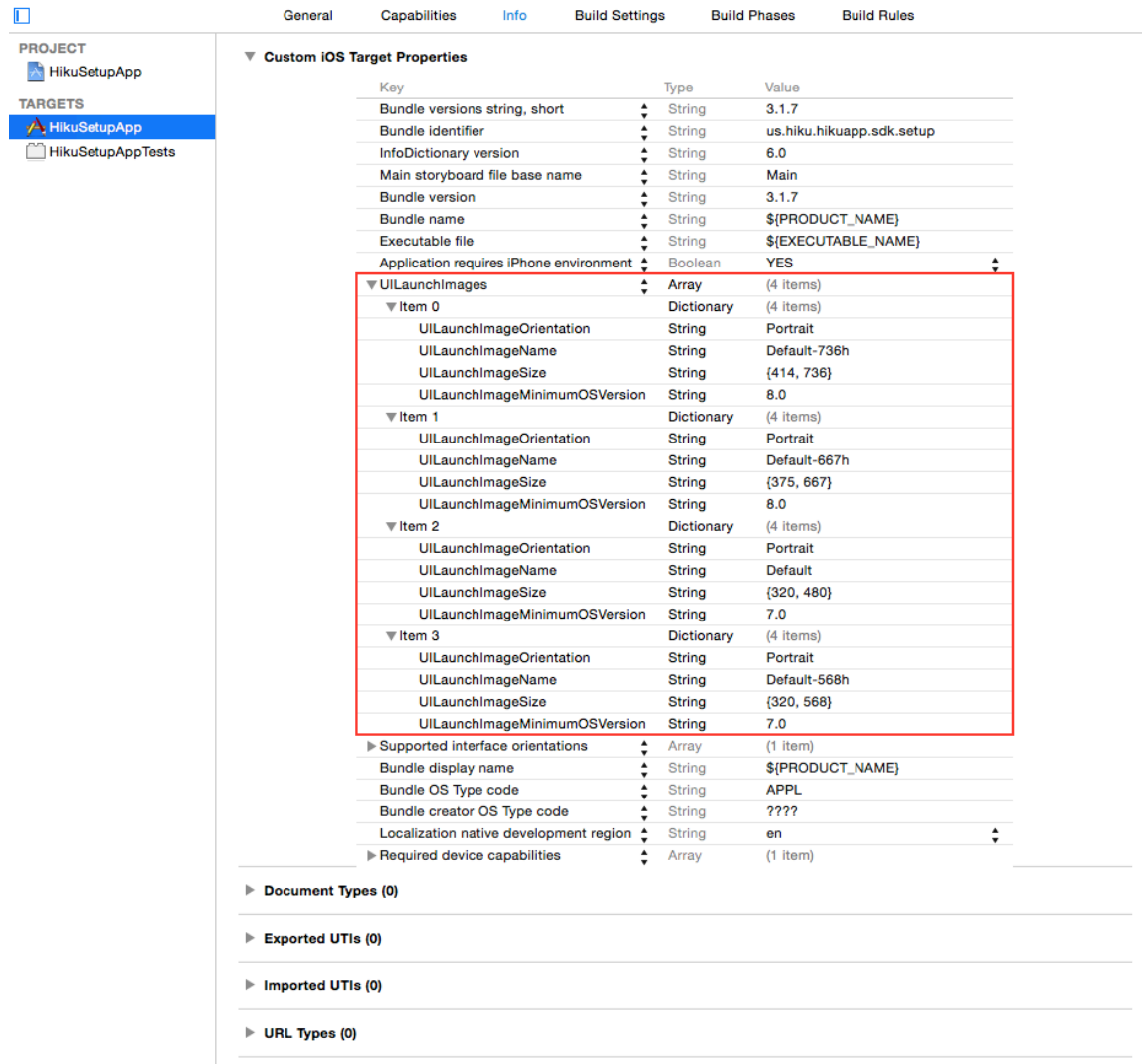
A demo app named `HikuSetupApp` has been provided as a proof-of-concept and showcases how best to integrate with the SDK. The included `HKSetupSDKFramework.framework` has everything necessary to integrate with hiku, including a resource bundle with all graphics, custom fonts, and audio tones.

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:



The screenshot shows the Xcode interface with the 'Info' tab selected for a custom iOS target. The 'UILaunchImages' array is expanded, showing four items (Item 0 to Item 3). Each item is a dictionary with four keys: 'UILaunchImageOrientation' (Portrait), 'UILaunchImageName' (Default-736h, Default-667h, Default, Default-568h), 'UILaunchImageSize' ({414, 736}, {375, 667}, {320, 480}, {320, 568}), and 'UILaunchImageMinimumOSVersion' (8.0, 8.0, 7.0, 7.0). The array is highlighted with a red box.

Key	Type	Value
Bundle versions string, short	String	3.1.7
Bundle identifier	String	us.hiku.hikuapp.sdk.setup
InfoDictionary version	String	6.0
Main storyboard file base name	String	Main
Bundle version	String	3.1.7
Bundle name	String	\$(PRODUCT_NAME)
Executable file	String	\$(EXECUTABLE_NAME)
Application requires iPhone environment	Boolean	YES
UILaunchImages	Array	(4 items)
Item 0	Dictionary	(4 items)
UILaunchImageOrientation	String	Portrait
UILaunchImageName	String	Default-736h
UILaunchImageSize	String	{414, 736}
UILaunchImageMinimumOSVersion	String	8.0
Item 1	Dictionary	(4 items)
UILaunchImageOrientation	String	Portrait
UILaunchImageName	String	Default-667h
UILaunchImageSize	String	{375, 667}
UILaunchImageMinimumOSVersion	String	8.0
Item 2	Dictionary	(4 items)
UILaunchImageOrientation	String	Portrait
UILaunchImageName	String	Default
UILaunchImageSize	String	{320, 480}
UILaunchImageMinimumOSVersion	String	7.0
Item 3	Dictionary	(4 items)
UILaunchImageOrientation	String	Portrait
UILaunchImageName	String	Default-568h
UILaunchImageSize	String	{320, 568}
UILaunchImageMinimumOSVersion	String	7.0
Supported interface orientations	Array	(1 item)
Bundle display name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle creator OS Type code	String	????
Localization native development region	String	en
Required device capabilities	Array	(1 item)

The file “HikuSetupApp/HikuSetupApp/HikuSetupApp-Info.plist” can be referenced for the required values. **Note that no images actually have to be added, only the property strings mentioned above.**

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:

▼ **App Icons and Launch Images**

App Icons Source	AppIcon
Launch Images Source	LaunchImage
Launch Screen File	

8.0 Known Issues

The following are known issues in the hiku SDK:

8.1 In-call Status Bar Issues

Works on: iOS 7.x

Doesn't work on: iOS 8.x

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.