Encanta Developer's Guide (iOS)

Contents

---

1. Create a new app in the Encanta web portal

    a. Go to http://app.getencanta.com
    b. Sign in, or create an account
    c. On the Your Apps page, click +NEW to create a new app
    d. Enter the name of your app and click Create

2. Add Encanta support to your iOS app

    a. Add the Encanta Framework

        i. Add Encanta Framework and Dependencies - using Cocoapods (https://cocoapods.org)

        Coming soon!

        ii. Add the Encanta Framework and Dependencies - not Using Cocoapods

            1. Add the SDK to your XCode project by dragging both Encanta.framework and Encanta.bundle into your project.
            2. In your Xcode project's Build Settings, add –ObjC to Other Linker Flags.
            3. Download the current version of AFNetworking from github (https://github.com/AFNetworking/AFNetworking)
            4. Follow the instructions for adding AFNetworking to your project, either manually, by adding the source files to your Xcode project, or by using Cocoapods.

    b. Add System Dependencies

        i. In your Xcode project under Build Phases -> Link Binary With Libraries, add the following system frameworks and libraries:

            1. AssetsLibrary.framework
            2. SystemConfiguration.framework
            3. libicucore.dylib
            4. libc++.dylib

3. Basic Functions

a. Register your app with Encanta

To register your app with Encanta, add the following line of code to your app:

Objective-C:

```
[Encanta sharedInstanceWithAppToken:@"YOUR_ENCANTA_APP_TOKEN"];
```

Swift:

```
Encanta.sharedInstanceWithAppToken("YOUR_ENCANTA_APP_TOKEN")
```

where `YOUR_ENCANTA_APP_TOKEN` is the unique token for your application, displayed under App Settings in the Encanta web portal.

The best place to add the above code is in your class that implements the `UIApplicationDelegate protocol`, in the `application:didFinishLaunchingWithOptions:` method.

Add the following line of code to the imports section (if you'll be calling Encanta APIs from your Swift code, add this line to your bridging header):

```
#import <Encanta/Encanta.h>
```

b. Show the Encanta messaging UI

To show the Encanta messaging UI, paste the following line of code at the desired location:

Objective-C:

```
[self.navigationController
|          pushEncantaViewControllerWithTitle:@"<view title>"
                                      animated:YES];
```

Swift:

```
self.navigationController!.pushEncantaViewControllerWithTitle(
          "<view title>", animated:true)
```

Again, for Objective-C code, be sure to add the following to the imports section of the implementation file:

```
#import <Encanta/Encanta.h>
```

Alternatively, to present the Encanta messaging UI modally:

Objective-C:

```
[self presentEncantaViewControllerWithTitle: @"<view title>"
                                    animated:YES completion:^{
}];
```

Swift:

```
self.presentEncantaViewControllerWithTitle("<view title>",
                                animated: true) { () -> Void in
}
```

c. Handle incoming messages notification

When a new message is received (from the web portal), the Encanta framework posts a notification via `NSNotificationCenter`. To register for this notification:

Objective-C:

```
[[NSNotificationCenter defaultCenter]
           addObserver:self
              selector:@selector(handleNewSupportChatNotification:)
                  name:kEncantaChatReceivedNotification
                object:nil];
```

Swift:

```
NSNotificationCenter.defaultCenter().addObserver(self,
      selector:"handleNewSupportChatNotification:",
          name:kEncantaChatReceivedNotification,
        object: nil)
```

And then handle the notification as desired:

Objective-C:

```
/* handle notification */
- (void) handleNewSupportChatNotification:(NSNotification*) notification {

    /* show messaging UI? */
}
```

Swift:

```
/* handle notification */
@objc func handleNewSupportChatNotification(
                         notification: NSNotification){
    /* show messaging UI? */
}
```

d. Set the customer's name

By default, your users will appear in the Encanta web portal and identified using auto-generated names (e.g., "User 951"). You can provide an actual name as follows:

Objective-C:

```
[[Encanta sharedInstance] setUserName:@"<user name>"];
```

Swift:

```
Encanta.sharedInstance().setUserName("<user name>")
```

The web portal will then display the user's name instead of the auto-generated name.

4. Advanced Functions

   a. Get the Encanta version

   Retrieve the version string for the Encanta framework via the `version` property:

   Objective-C:

   ```
   NSString* encantaVersion = [Encanta sharedInstance].encantaVersion;
   ```

   Swift:

   ```
   let encantaVersion = Encanta.sharedInstance().encantaVersion
   ```

   b. Retrieve the unread message count

   Retrieve the number of unread Encanta messages via the `unreadCount` property:

   Objective-C:

   ```
   NSUInteger numUnread = [Encanta sharedInstance].unreadCount;
   ```

   Swift:

   ```
   let numUnread = Encanta.sharedInstance().unreadCount
   ```

   This property is KVO-compliant, so your can register your class as a KVO observer of this property and be notified when the unread count changes.

   c. Determine if messaging is enabled

   In-app messaging can be enabled or disabled via the web portal. When in-app messaging is disabled, you may wish to hide any messaging-related UI in your application. Retrieve the status of in-app messaging via the `chatEnabled` property:

   Objective-C:

   ```
   BOOL messagingEnabled = [Encanta sharedInstance].chatEnabled;
   ```

   Swift:

   ```
   let messagingEnabled = Encanta.sharedInstance().chatEnabled
   ```

   This property is KVO-compliant, so your can register your class as a KVO observer of this

property and be notified when the flag changes.

d. Set custom properties

The Encanta web portal displays a number of properties for each user running a registered app. This default property set includes:

i. Application version: the version of your app
ii. Sign-up Date: the date when the user first starting using your app
iii. App Opens: the number of times the user opens your app
iv. Last Seen: how long ago did the user last use your app

In addition to this default set of properties, Encanta allows your app to set any number of custom properties. Each property consists of a name and a value. To set a property, use the following API:

Objective-C:

```
[[Encanta sharedInstance] setPropertyWithName:@"<property name>"
                                        value:<value>];
```

Swift:

```
Encanta.sharedInstance().setPropertyWithName("<property name>",
                                        value:<value>)
```

The above API currently supports the following types for `<value>`: `NSString`, `NSNumber`, and `NSDate`.

To increment a numeric property, use the following API:

Objective-C:

```
[[Encanta sharedInstance] incrementPropertyWithName:@"<property name>"];
```

Swift:

```
Encanta.sharedInstance().incrementPropertyWithName("<property name>")
```

e. Enable push notifications

When a user receives a new Encanta message from the web portal, the Encanta server can send your app a push notification via the Apple Push Notification Service (APNS). To enable push notifications, you first need to configure push notifications for your app (https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringPushNotifications/ConfiguringPushNotifications.html).

Next, you need to export your Apple Push certificate from your Keychain, and then upload the resulting .p12 file to Encanta. The latter is done via the Encanta web portal (http://app.getencanta.com).

Finally, at runtime, your app needs to provide the device token to Encanta by calling the following API:

Objective-C:

```
[[Encanta sharedInstance] registerPushToken:@"<device token>"];
```

Swift:

```
Encanta.sharedInstance().registerPushToken("<device token>")
```

The device token is provided by iOS, after your app has asked the user for permission to receive push notifications.

For more information on how to request for permission to receive push notifications and handling push notifications sent to your app, consult the Apple developer documentation (https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG).

f. Customize the look of the messaging UI

The Encanta framework supports a number of properties which enable customization of a number of user interface elements in the messaging UI:

Property Name	Description
viewBackgroundColor	Sets/gets the color of the background view of the Encanta messaging UI (default = white)
incomingMessageBubbleColor	Sets/gets the color of the chat message "bubble" for incoming messages from the web portal (default = dark gray)
outgoingMessageBubbleColor	Sets/gets the color of the chat message "bubble" for outgoing messages (default = blue/green)
incomingMessageTextColor	Sets/gets the color of the chat message text for incoming messages (default = white)
outgoingMessageTextColor	Sets/gets the color of the chat message text for outgoing messages (default = black)
messageTextFont	Sets/gets the font for the chat message text, both incoming and outgoing (default = system font, 15 pt)

incomingTimestampTextColor	Sets/gets the color of the timestamp text for incoming messages (default = light gray)
outgoingTimestampTextColor	Sets/gets the color of the timestamp text for outgoing messages (default = light gray)
timestampTextFont	Sets/gets the font for the timestamp text, both incoming and outgoing (default = system font, 11 pt)
usesDynamicType	Sets/gets a flag which determines whether or not the messaging UI components support Dynamic Type (default = YES)

For example, to set the background color of the Encanta messaging view to light gray:

Objective-C:

```
[[Encanta sharedInstance].viewBackgroundColor = [UIColor lightGrayColor]
```

Swift:

```
Encanta.sharedInstance().viewBackgroundColor = UIColor.lightGrayColor()
```