Legacy - iOS Quick Start Guide

This documentation will help you quickly start using Alchemer Mobile in your iOS app. For complete documentation, as well as system requirements, see the iOS Integration guide.

1. Add Alchemer Mobile

There are several options for adding the Alchemer Mobile (formerly Apptentive) SDK to your app. The easiest method is to use CocoaPods. If you would like to use another integration method, please see our iOS Integration guide.

CocoaPods

You will need a recent version of CocoaPods, a text editor, and basic familiarity with the Terminal app. If you aren't already using CocoaPods for your app, start by opening up the Terminal app and navigating to the directory that contains your app's `.xcodeproj` file. Then run `pod init` to create a **Podfile** for your app.

Open your app's Podfile with your favorite text editor, and add an entry for Alchemer Mobile:

```
 # Uncomment this line to define a global platform for your project
# platform :ios, '9.0'

target 'MyApp' do
  # Comment this line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for MyApp
  pod 'apptentive-ios'
end
```

Then, in the same directory as your Podfile, run `pod install` in the Terminal app. Finish by opening up the `.xcworkspace` file created by CocoaPods.

2. Initialize Alchemer Mobile

When your app starts, it will need to initialize the Alchemer Mobile SDK.

First, you'll need to import the Alchemer Mobile SDK, and then create a configuration object with your Alchemer Mobile (Apptentive) App Key and Alchemer Mobile (Apptentive) App Signature. Register Alchemer Mobile with that configuration object. Finally, set your app's iTunes Store ID. We recommend doing this in your application delegate's `application(_:didFinishLaunchingWithOptions:)` method:

```
 import UIKit
import Apptentive

class AppDelegate: UIResponder, UIApplicationDelegate {
var window: UIWindow?

  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationL
aunchOptionsKey: Any]?) -> Bool {
    if let configuration = ApptentiveConfiguration(apptentiveKey: "<#Your Apptentive App Key#>
", apptentiveSignature: "<#Your Apptentive App Signature#>") {
        configuration.appID = "<#Your iTunes App ID#>"
        Apptentive.register(with: configuration)
    }

    // Other app initialization...

    return true
  }
}
```

> Make sure you use the Alchemer Mobile (formerly Apptentive) App Key and Alchemer Mobile (formerly Apptentive) App Signature for the iOS app you created in the Alchemer Mobile console. Sharing these credentials in two apps, or using ones from the wrong platform is not supported, and will lead to incorrect behavior.

3. Add Events

Events record user interaction. You can use them to determine if and when an Interaction will be shown to your customer. At a minimum, you should include 20-50 Events in your app to start taking advantage of Alchemer Mobile, but for now, let's just create one. To trigger an Event, call the `engage()` method. This will record the Event, and then check to see if any Interactions targeted to that Event are allowed to be displayed, based on the logic you set up in the Alchemer Mobile Dashboard.

In this example, trigger an Event when your Main Activity resumes.

```
 import UIKit
import Apptentive

class MainViewController: UIViewController {

  // ...

  override func viewDidAppear(animated: Bool) {
    super.viewDidAppear(animated: animated)

    Apptentive.shared.engage(event: "main_view_appeared", from: self)
  }
}
```

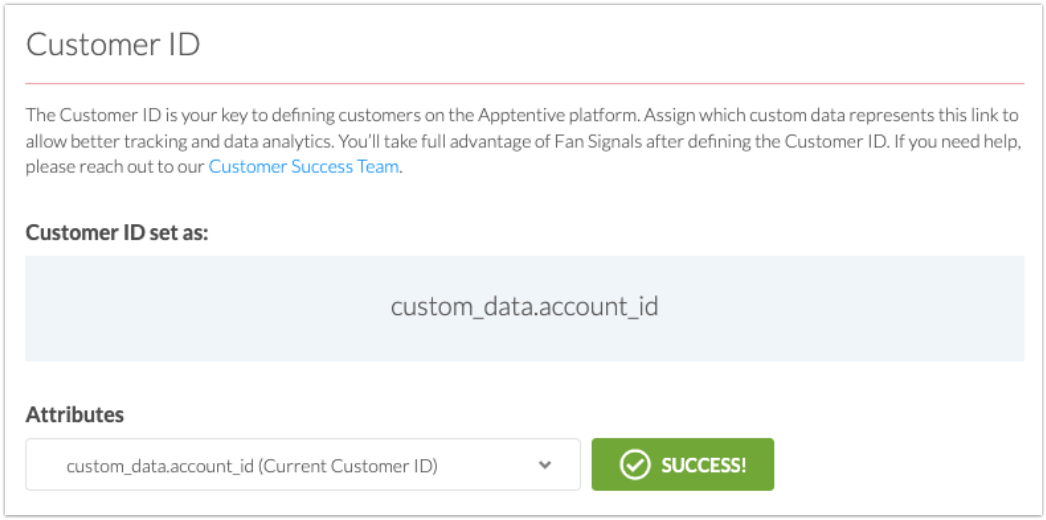
4. Add Customer ID

You can send Custom Data associated with a person's profile that is using the app, or the device. In particular, this is useful for sending a Customer ID and other information that helps you understand and support your users better. Custom Data can also be used for configuring when Interactions will run. You can add custom data of type `String`, `Number`, and `Boolean`.

Below is an example of a Customer ID value being passed in, along with whether that customer is on a premium account of the app or not.

```
Apptentive.shared.addCustomPersonData("1234321", withKey: "CustomerID")
Apptentive.shared.addCustomPersonData(true, withKey: "is_premium")
```

After setting your Customer ID and other custom data, you can choose which field is your Customer ID in the Alchemer Mobile Platform.



5. Show a Survey

Now that you've created an Event, you can create a Survey and display it when the Event is triggered.

1. Go to the Surveys page.
2. Click "New Survey" to create a new survey.
3. Give the Survey a name, title, introduction, add a question, choose whether to end with a *Thank You* message, and click **Save & Continue**.
4. Choose *Publish survey as an independent Interaction*.
5. Under the **Where** section, chose the Event `main_view_appeared` (or whatever Event name you used). If your app hasn't connected to the server after triggering that Event, you will need to add it manually at this point, by clicking **Create new Event** on the Events page.

6. Near the bottom, check ***Allow multiple responses from the same person*** so you can display this survey more than once.
7. Click **Save & Continue.**
8. Click **Launch Survey.**
9. Finally, **uninstall then reinstall** the app to ensure you have downloaded that newly launched Survey from our servers.

Now, you will see this survey when you trigger the `main_view_appeared` Event.

6. Add Message Center

Find a place in your app for a button that will launch Message Center. This will allow customers to contact you with feedback, or questions if they are having trouble using your app, as well as allow them to see your responses.

If Message Center is available, show a `UIButton` or `UITableViewCell` that will launch it when tapped. This example assumes you have an `UIViewController` subclass called `SettingsViewController` that has a `UIButton` you would like to open Message Center with.

```
 import UIKit
import Apptentive

class SettingsViewController: UIViewController {

  // ...

  @IBAction func openMessageCenter(sender: UIButton) {
    Apptentive.shared.presentMessageCenter(from: self)
  }
}
```

Related Articles