Alchemer Mobile iOS SDK Migration Guide

This guide is intended for developers who have been using the Alchemer Mobile (formerly Apptentive) iOS SDK and are migrating to the version 6 of the SDK.

Prerequisites

- Your app must have a deployment target of iOS 11 or later
- For the existing version's data to be migrated, your existing Alchemer Mobile SDK (formerly Apptentive) must be version 4 or later.
- A handful of features are currently unavailable (notably client authentication), and if using them is a requirement for your app, you may wish to delay your migration until replacements are in place. Please see the Release Notes for more information.
- We recommend calling all Alchemer Mobile (formerly Apptentive) methods on the main queue.

Quick Overview Video

Your browser does not support HTML5 video.

View how simple it is to migrate to ApptentiveKit (6.0 and up)

Update the Alchemer Mobile Dependency

Replace with Swift Package Manager

You can now use the built-in package management features of Xcode to add ApptentiveKit to your app.

Choose File > Add Packages... and enter `https://github.com/apptentive/apptentive-kit-ios` in the search field.

Don't forget to remove the Alchemer Mobile dependency from wherever you were previously including it.

Update CocoaPods

Open your app's Podfile with your favorite text editor, and modify the entry for Alchemer Mobile (ApptentiveKit) as follows:

```
 source 'https://github.com/apptentive/cocoapods-specs.git'
platform :ios, '12' # Minimum deployment target is 11
use_frameworks!

target 'My App' do
   # Pods for MyApp
   pod 'ApptentiveKit', '~> 6.0'

      # Be sure to remove or comment out the 'apptentive-ios' pod
      # pod 'apptentive-ios'
end
```

Then, in the same directory as your Podfile, run `pod install` in the Terminal app.

The new SDK requires an iOS deployment target of 11.0 and above, so please ensure your project (and Podfile) is configured for that.

> *Please note that both old and new SDK pods being present may cause the app to crash.*

Update Subproject

To update the Alchemer Mobile (ApptentiveKit) dependency as a subproject, start by removing the old Alchemer Mobile (ApptentiveKit) subproject, and then clone or download the code from https://github.com/apptentive/apptentive-kit-ios, and drag the `ApptentiveKit.xcodeproj` file into your app project.

Update Framework

Download the latest ApptentiveKit framework from https://github.com/apptentive/apptentive-kit-ios/releases, and use it to replace the existing version of Apptentive.xcframework in your app.

Update Carthage

Update your Cartfile to remove the previous entry (`github "apptentive/apptentive-ios" >= 5.0.0`) and add `github "apptentive/apptentive-kit-ios" >= 6.0.0` . Then run `carthage update --use-xcframeworks` . Don't forget to remove the old dependency in your target's *General* tab in the *Frameworks, Libraries, and Embedded Content* section and drag the `ApptentiveKit.xcframework` folder into that section from the `Carthage/Build` folder.

Update `import` Statements

Within your project, modify all `import Apptentive` instances to `import ApptentiveKit` within each source file in which you plan to use the Alchemer Mobile SDK.

After this step, you should be able to build your app successfully using the new SDK, but you will likely see some deprecation warnings.

Fix Deprecation Warnings

Registering the SDK

The `register(with:)` method is now an instance method, and has been modified to accept an `Apptentive.AppCredentials` object as its first argument. The optional second argument is a completion handler that is called when the SDK registration completes or fails.

```
import ApptentiveKit

func application(_ application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
Apptentive.shared.register(with: .init(key: "<#Your Apptentive App Key#>", signature: "<#Your Apptentive App Signature#>"))

    return true
}
```

Engaging Events

If you trigger events now using a string literal, you can continue to pass it exactly as you do now without any changes (example below).

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

If you're using a variable to store the event name, the engage now accepts an `Event` object as its first argument, so you'll need to explicitly initialize it as follows:

```
Apptentive.shared.engage(event: Event(named: eventName), from: self)
```

Engaging events with custom data is currently unavailable.

Support for engaging events with extended data has been permanently removed.

Person and Device Custom Data

The previous `addCustomPersonData(_:withKey:)` and `removeCustomPersonData(_:withKey:)` (and the corresponding methods for device custom data) have been deprecated when used in Swift.

Instead, use subscripting the `personCustomData` and `deviceCustomData` properties.

Because these properties are a `struct` type, you must continue to use the older methods from Objective-C.

Notifications

The individual notifications emitted by previous versions of the SDK have been replaced with a single new notification: `Notification.Name.apptentiveEventEngaged`.

Your app can listen to this notification and then examine the values in the `userInfo` dictionary for the following keys:

- `eventType`: the extended name of the event, for example `com.apptentive#Survey#submit`

- `interactionType` : the type of the interaction that engaged the event, or `app` if not applicable
- `interactionID` : the internal identifier of the interaction that engaged the event, if applicable
- `eventSource` : the source of the event, either `com.apptentive` (for events that are engaged by the SDK itself) or `local.app` (for events that are engaged by your app)

For a complete list of internal events, see the integration guide.

Customer Authentication

The method signatures for customer authentication have been updated for Swift developers. The completion handlers for `logIn(with:completion:)` and `updateToken(_:completion:)` now accept a `Result<Void, Error>` parameter.

Also the authentication failure callback has been replaced with a delegate protocol (`ApptentiveDelegate`) that calls `authenticationDidFail(with:)` the first time an authentication failure occurs when sending a background request (such as an event, message, or survey response).

Customization

Customizing the styling of Alchemer Mobile UI elements has changed significantly. The `ApptentiveStyle` protocol and `ApptentiveStyleSheet` class have been removed (although placeholders are available to allow your code to build with warnings rather than errors).

Themes

The SDK will ordinarily apply a styling theme using Alchemer Mobile's default colors alongside system fonts and symbols. This theme is designed to look consistent across our iOS and Android SDKs, but may differ from your app's look and feel and the rest of the iOS system.

By setting the Alchemer Mobile class's `theme` property to `.none` (note: this must be done before calling the `register(with:completion` method), the interactions take on a more native iOS look and feel, but it will differ substantially from the Android SDK.

UIAppearance

With the use of the `.none` theme, the ApptentiveKit interaction UI will pick up certain UIAppearance overrides that your app has set (for example, navigation bar tint).

If an appearance setting in your app makes ApptentiveKit interactions unattractive or unreadable, you can use `appearance(whenContainedInInstancesOf: [ApptentiveNavigationController.self])` to make a corrective appearance change to ApptentiveKit interactions (specifically those, like Message Center and Surveys, that use view controllers other than UIAlertController).

UIKit Extensions

For styling changes that aren't amenable to UIAppearance, ApptentiveKit defines a number of extensions to common UIKit classes to allow setting colors, fonts, images, and more. For more information, see our forthcoming Customization Guide.

Overriding InteractionPresenter

For particularly extensive customizations, we recommend subclassing the InteractionPresenter class and setting your subclass as the value for Alchemer Mobile's `interactionPresenter` property. You can override the methods that present each interaction type, instantiating your own view controllers and presenting them as you see fit. A view model for each interaction is provided to configure your view controllers for displaying the interaction and reacting to user input. More information on this technique will be added soon.

Data that will be Migrated

The following data will be automatically migrated the first time that the new version of the SDK is initialized, provided that the previously-integrated Alchemer Mobile's SDK was version 4 or newer:

- Conversation credentials (identifier and token)
- The number of times each event was engaged and when it was last engaged
- The number of times each interaction was presented and when it was last presented
- Person name, email address, and custom data
- Device custom data
- Random sampling data

The following data will not be migrated:

- Events, messages, and survey responses that could not be sent because the device was offline
- The local cache of messages and attachments
- Any data from SDK versions prior to 4.0

Related Articles