Legacy - Migrating Alchemer Mobile iOS Versions

Certain versions of our SDK break compatibility with older versions. If you are migrating from an old version, please follow the migration guides below for each version between the version you are using and the version you are migrating to.

v5.0.0

If you have integrated a previous version of the Alchemer Mobile SDK, you will need to keep in mind the following changes in our version 5.0.0 release. For more information, please see our Integration Reference.

New Asynchronous `engage` Methods

The `engage` methods no longer return a boolean value indicating whether an interaction was presented in response to the event being engaged. Instead, there are versions of each method that accept a completion handler that will be called with that result.

For example, where previously you might have done the following:

```
let didShowMessageCenter = Apptentive.shared.presentMessageCenter(from: self)

if !didShowMessageCenter {
    print("message center was not shown")
}
```

You would now write:

```
Apptentive.shared.presentMessageCenter(from: self.window!.rootViewController!) { (success) in
    if !success {
        print("message center was not shown")
    }
}
```

Likewise the `canShowInteraction(forEvent:)` and `canShowMessageCenter()` methods were replaced by `queryCanShowInteraction(forEvent:,completion:)` and `queryCanShowMessageCenter(completion:)`. For example, to enable a Message Center button if Message Center is available, you could use code like the following:

```
Apptentive.shared.queryCanShowMessageCenter { (canShow) in
    if canShow {
        // enable message center button
    }
}
```

New `UNUserNotificationCenter` methods

You can use the User Notifications framework in your apps that target iOS 10 and later.

If your app does not use push or local notifications for purposes other than Alchemer Mobile, you can simply set the current user notification center's delegate property to the Alchemer Mobile singleton:

```
UNUserNotificationCenter.current().delegate = Apptentive.shared
```

Alternative, if your app needs to respond to non-Alchemer Mobile push and local notifications, you can forward them as follows:

```
 func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response: UNNotificationRespons
e, withCompletionHandler completionHandler: @escaping () -> Void) {
    // Pass in a view controller, or nil to have Apptentive create a new window for Message Center
    let handledByApptentive = Apptentive.shared.didReceveUserNotificationResponse(response, from: viewCo
ntroller, withCompletionHandler: completionHandler)

    if (!handledByApptentive) {
        // Handle the notification
        completionHandler()
    }
}

func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification: UNNotification, withC
ompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {
    let handledByApptentive = Apptentive.shared.willPresent(notification, withCompletionHandler: completion
Handler)

    if (!handledByApptentive) {
        // Decide how to present the notification
        completionHandler(.alert)
    }
}
```

Please note that in both cases you will still need to forward remote notifications to Alchemer Mobile for Alchemer Mobile push to work.

v4.0.0

If you have integrated a previous version of the Alchemer Mobile SDK, you will need to keep in mind the following changes in our version 4.0.0 release. For more information, please see our Integration Reference.

Moved from Static Library to Dynamic Framework

We still recommend integrating using CocoaPods. If you previously integrated using a static library, it has been removed from the project and replaced with a dynamic framework.

The easiest way to use this framework is by using Carthage.

New SDK Registration Process

In place of the a single API key, the SDK now uses a key and signature. These are available from the same section of your Alchemer Mobile dashboard where you previously found your API key. You use these to create an instance of the new ApptentiveConfiguration class.

You should pass this configuration instance to the register(with:) class method on the Alchemer Mobile class.

After that you can use the SDK as you normally would.

New Login/Logout Feature

A new logIn(withToken:completion:) method has been added, along with a corresponding logOut method. These allow multiple users to use the same app instance without being exposed to one another's messages and custom data.

The logIn method takes a JSON Web Token that you can generate on your server when a user authenticates using your app. The JWT signing secret for your app can be found in the same section of your Alchemer Mobile dashboard as the key and signature.

You should also set the authenticationFailureCallback property on the Alchemer Mobile singleton so that your app can be notified and reauthenticate in case the token is revoked or has expired.

Runtime Log Level Setting

You can now set the log level at runtime. The easiest way to do this is to set the logLevel property on the configuration object before you register the SDK. The setting defaults to INFO for all build configurations.

You can also set the logLevel property directly on the Alchemer Mobile singleton after you have registered the SDK.

Local Notification Forwarding for Push Notifications

To support multiple users without exposing potentially sensitive messages in notifications, the SDK uses a two step process to implement push notifications. A silent push is sent from the server, and if the intended recipient is logged in, a local notification is posted.

When the user responds to this local notification, your app should forward it to the Alchemer Mobile SDK using the new didReceiveLocalNotification(_:from:) method. The first parameter is the local notification passed into your app delegate’s application(_ application:didReceive:) method (which your app delegate must implement), and the second is a view controller suitable for presenting the Message Center view controller from.

v3.0.0

If you have integrated a previous version of the Alchemer Mobile SDK, you will need to keep in mind the following changes in our version 3.0.0 release. For more information, please see our Integration Reference.

Major Changes

Survey Redesign

The surveys provided by the Alchemer Mobile SDK have been extensively redesigned, although their functionality remains the same.

Style Sheet Has Been added

A new style sheet property has been added to the `Apptentive` singleton that greatly expands your ability to customize the Alchemer Mobile UI. You can use the default `ApptentiveStyleSheet` instance, or create your own, either by subclassing or by implementing the `ApptentiveStyle` protocol.

Currently the style sheet is used to apply styles to Surveys and the Message Center.

You will need to import the `ApptentiveStyleSheet.h` file if you would like to use the built-in styles and you are integrating via source or using the static library.

You can find more information in our iOS Interface Customization guide.

Renamed Classes and Constants

To avoid a namespace collision with a private iOS system framework, classes that previously used an `AT` prefix now use an `Apptentive` prefix. Additionally `ATConnect` has been renamed to simply `Apptentive`. Compatibility aliases have been added for the `ATConnect` and `ATNavigationController` classes.

Additionally a number of constants have had their names change from using an `AT` prefix to using an `Apptentive` prefix. The push provider and notification names are most likely to require updating in your code.

Renamed APIKey Property

The previous `apiKey` property has been renamed to `APIKey` to better follow Apple's naming convention. The previous capitalization is provided for compatibility, but has been deprecated.

v2.0.0

Major Changes

iOS Version Support

Alchemer Mobile SDK version 2.0.0 has a deployment target of iOS 7.0, which will support iOS 7, 8, and 9. In the 2.0.0 release we have dropped support for iOS 5 and 6.

Message Center

Message Center has been completely redesigned to improve its appearance and performance. Please ensure that the new Message Center UI displays properly in your app.

Message Center is still presented via the `presentMessageCenterFromViewController:` method.

Feedback Dialog has been Removed

The Feedback Dialog one-way message tool has been removed in favor of simply displaying Message Center.

In previous versions, people used the Feedback Dialog to submit their first message. Thereafter, they were sent to Message Center to read replies or send additional feedback.

There was formerly an option to disable Message Center and **only** accept messages via the one-way Feedback Dialog. This option has been removed.

Rather than using one-way messages via the Feedback Dialog, you should use a custom **Status Message** in the Ratings Prompt Configuration to set proper expectations of reply.

Message Center is Retrieved from Server

Message Center text is now sent to devices from the Alchemer Mobile backend. Much of this text is editable on a per-app basis via your Alchemer Mobile dashboard. These remote strings allow you to customize Message Center copy and localization at any point without issuing an app update.

As a consequence, we the SDK will be unable to show Message Center until that device syncs at least one time with the Alchemer Mobile servers. This sync should normally happen very quickly after the very first launch of the app.

If the first sync has not yet occurred, Alchemer Mobile displays a “We’re attempting to connect” message rather than the (unavailable) Message Center. This view will be seen only rarely in the actual usage of your app, but do be aware that you may see it in development if you try to launch Message Center immediately after a fresh install.

The new API method `canShowMessageCenter` has been added to indicate whether Message Center has been synced and can be displayed. If that method returns `NO` you can, for example, hide the *Message Center* button in your interface.

Name and Email Address Properties

The API for adding Name and Email Address details has been simplified.

Setting the user’s email and name using the new new `personName` or `personEmailAddress` properties on `ATConnect` will send that information to our server, and it will display alongside

messages that user sends in the Conversations View.

Please be aware that setting `personName` or `personEmailAddress` will **immediately overwrite** anything the person had previously typed in those fields, so you might want to check their values first. The person using your app will be given the opportunity to change those details. However, setting the properties programmatically again will overwrite the user-inputted values.

We have also removed the `initialUserName` and `initialUserEmailAddress` properties to simplify the API.

Push Notifications

The new method `setPushNotificationIntegration:withDeviceToken:` has been added to add a single Push Notification provider. To register for push notifications, call this method with one of the enumerated `ATPushProvider` values, plus the device token from `application:didRegisterForRemoteNotificationsWithDeviceToken`.

In light of this new method, we have removed the legacy integration API methods:

- `addIntegration:withConfiguration:`
- `addIntegration:withDeviceToken:`
- `removeIntegration:`
- `addApptentiveIntegrationWithDeviceToken:`
- `addUrbanAirshipIntegrationWithDeviceToken:`
- `addAmazonSNSIntegrationWithDeviceToken:`
- `addParseIntegrationWithDeviceToken:`

Alchemer Mobile Push Notifications will, if possible, now trigger a message fetch in the background. To enable background fetch, several API and project changes are needed:

- To enable Message Center background fetch, you should use the `fetchCompletionHandler:` versions of `didReceiveRemoteNotification:` on the App Delegate and on `ATConnect`.
- To enable Message Center background fetch, your app must set Remote Notifications as a valid Background Mode. This mode can be enabled in Xcode via your Target's Capabilities tab, or by adding the value `remote-notification` as a `UIBackgroundMode` in your app's `Info.plist`.
- A `BOOL` return type has been added to the `ATConnect` `didReceiveRemoteNotification:` methods. The return value indicates if the Push Notification was sent by Alchemer Mobile.
- The `completionHandler` block will be called by Alchemer Mobile when the message fetch is completed. To ensure that messages can be retrieved, please do not call the `completionHandler` block yourself if the notification was sent by Alchemer Mobile.
- If the Push Notification was **not** sent by Alchemer Mobile (formerly Apptentive), the parent app is responsible for calling the `completionHandler` block.

Removed Legacy Properties

We have removed the `useMessageCenter`, `initiallyUseMessageCenter`, and `initiallyHideBranding` properties from the API. Please make sure to update your code if you are

setting any of these properties.

Using Message Center and Hiding Branding are now set via the configuration on your Alchemer Mobile dashboard.

Determining if Interactions will be Shown

The method `willShowInteractionForEvent:` has been marked as deprecated and renamed to `canShowInteractionForEvent:`. This terminology matches the new API method `canShowMessageCenter:`.

v1.5.0

This version is no longer supported. Please remove your Alchemer Mobile integration and re-integrate with the latest iOS Integration Reference.

Related Articles