Alchemer Mobile iOS 7.0 SDK Migration Guide

Alchemer Mobile iOS SDK migration guide

This guide walks you through migrating from **Alchemer Mobile (ApptentiveKit) iOS SDK version 6** to **version 7**. It's intended for developers who already have the SDK implemented and want to take advantage of the latest platform updates, customization options, and concurrency support.

If you're migrating from **version 5 or earlier**, please refer to the previous migration guide before continuing.

What's new in version 7

Version 7 introduces several major updates focused on modern iOS development, design flexibility, and ease of customization.

Highlights

- **Swift structured concurrency support**
  The SDK now works seamlessly with async/await and Swift's concurrency model.
- **Updated UI for iOS 26**
  A refreshed interface complements iOS 26's new Liquid Glass design language.
- **Asset-based UI customization**
  Customize colors and images using Asset Catalogs—no code required.
- **Global font configuration**
  Set the font for all Alchemer Mobile UI elements with a single line of code.

Prerequisites

Before upgrading, make sure your app meets the following requirements:

- Your app's **deployment target must be iOS 15 or later**

- Your app must already be using **ApptentiveKit version 5 or later** for existing data to migrate automatically

- All public SDK methods are now marked `@MainActor`

    - If you're calling SDK methods from a non-main isolation context, you'll need to create a `Task` and use `await`

Important note for developers upgrading from version 5

If you plan to change the **Alchemer Mobile dashboard instance** (App Key or App Signature):

- Users **must first update to SDK version 6 or later**
- Changing the key/signature is **not supported** in versions prior to 6.2.0 or when migrating directly from earlier versions

---

Update the Alchemer Mobile dependency

You can update the SDK using any of the supported dependency managers below.

Swift Package Manager (recommended)

We recommend managing ApptentiveKit using Xcode's built-in Swift Package Manager support.

1. Select your project in the **Project Navigator**

2. Select the project entry in the sidebar

3. Open the **Package Dependencies** tab

4. Update **ApptentiveKit** from version 6.x to **7.0**

CocoaPods

> ⚠ CocoaPods is in the process of shutting down the trunk repo. We recommend migrating to another integration method when possible.

1. Open your app's `Podfile`

2. Update the ApptentiveKit entry:

```
 source 'https://github.com/apptentive/cocoapods-specs.git'
platform :ios, '15' # Minimum deployment target is 15
use_frameworks!

target 'My App' do
  pod 'ApptentiveKit', '~> 7.0'

  # Remove or comment out the legacy pod if present
  # pod 'apptentive-ios'
end
```

3. Run `pod install` from the same directory

Subproject integration

1. Clone or pull the latest code from `https://github.com/apptentive/apptentive-kit-ios`
2. Ensure `ApptentiveKit.xcodeproj` is included in your app project

Framework integration

1. Download the latest `ApptentiveKit.xcframework` from `https://github.com/apptentive/apptentive-kit-ios/releases`
2. Replace the existing framework in your app with the new version

---

Carthage

1. Update your `Cartfile`:

```
github "apptentive/apptentive-kit-ios" >= 7.0.0
```

2. Run:

```
carthage update --use-xcframeworks
```

API changes in version 7

Swift concurrency updates

All public APIs are now isolated to the **Main Actor**.

If your app calls SDK methods from another isolation context, wrap the call in a `Task` and use `await`.

Registering the SDK

The `register(with:)` method now includes an optional **region** parameter.

Supported regions:

- `.us` (default)
- `.eu`
- `.au`

Example:

```
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#>"
    ),
    region: .us
  )

  return true
}
```

Async/await support

All public methods that previously relied on completion handlers now have async alternatives. Methods that previously returned a `Result` via completion handlers are now marked `async throws`.

Async-capable methods include:

- `register(with:region:)`
- `engage(event:from:)`
- `presentMessageCenter(from:with:)`
- `canShowInteraction(event:)`
- `canShowMessageCenter()`
- `logIn(with:)`
- `logOut()`
- `updateToken(_)`

If your app uses async/await, these APIs provide a cleaner and more modern developer experience.

Customizing the Alchemer Mobile UI

Customization in version 7 builds on the flexibility introduced in version 6, with new options for iOS 26 and later.

Themes

Set the SDK theme **before calling** **`register`**.

- On **iOS 26 and later**, SDK 7 defaults to the `.customer` theme, designed to align with Liquid Glass
- On **iOS 18 and earlier**, the SDK continues to default to the legacy `.apptentive` theme

To enable Asset Catalog customization, use:

- `.customer`
- `.customerBasedOnApptentive`

Asset Catalog customization (new)

Version 7 introduces UI customization through Asset Catalogs.

1. In your app's main Asset Catalog, create a folder named **Apptentive**

2. Inside that folder, add groups for:

- `Dialog` (Prompts and Love Dialog)
- `Survey`
- `MessageCenter`

3. Add colors—and where supported, images—to customize each interaction type

4. This is the preferred customization method for iOS 26 and later.

Global font setter (new)

You can now configure fonts for all Alchemer Mobile UI elements with a single property:

```
Apptentive.fontName = "YourFontFamilyName"
```

If you need more granular control, continue using UIKit extensions.

UIAppearance support

- UIAppearance settings from version 6.2+ still apply to:
  - Prompts
  - Love Dialogs
- This applies only on **iOS 18 and earlier**
- For **iOS 26 and later**, use:

    - Asset Catalogs (recommended), or
    - UIKit extension properties

UIKit extensions

ApptentiveKit includes UIKit extensions for setting fonts, colors, images, and more on individual UI elements.

Use this approach when:

- Asset Catalog customization isn't granular enough, or
- You need per-element overrides

See the **Customization Guide** for details.

Advanced customization: InteractionPresenter

For deeply customized experiences, you can subclass `InteractionPresenter` and assign it to the SDK's `interactionPresenter` property.

This allows you to:

- Present your own view controllers
- Fully control interaction layout and presentation
- Use provided view models to respond to user input

Data migration details

The following data is **automatically migrated** the first time SDK 7 is initialized, as long as the previously integrated SDK was version 4 or later:

- Conversation credentials (identifier and token)
- Event engagement counts and timestamps
- Interaction presentation counts and timestamps
- Person name, email, and custom data
- Device custom data
- Random sampling data

Data that is not migrated

- Events, messages, or survey responses that were not sent due to the device being offline
- Local caches of messages and attachments
- Any data from SDK versions earlier than 4.0

Related Articles