Android Quick Start Guide

This guide helps you quickly start using Alchemer Mobile **6.0+** in your Android app. For complete documentation see the Android Integration guide.

System Requirements

`minSDK` version: 21 (Android 5.0)

`compileSDK` version: 31 to 33

Gradle version: >= 7.0.0

- `classpath 'com.android.tools.build:gradle:7.0.0'`

Set up the Alchemer Mobile SDK

To install the SDK, add `apptentive-kit-android` to the `dependencies` block of your `app/build.gradle` file, replace APPTENTIVE_VERSION with the most recent one.

```
 repositories {
  mavenCentral()
}

dependencies {
    implementation 'com.apptentive:apptentive-kit-android:APPTENTIVE_VERSION'
}
```

> For details on the latest SDK releases and past versions, see the Releases page on GitHub

Configure the SDK using `ApptentiveConfiguration` with your Alchemer Mobile **App Key** and **Alchemer Mobile App Signature** in your `Application` subclass. These values are available from the **API & Development** section of the **Settings** tab in your **Alchemer Mobile Dashboard**. For more information on the optional `ApptentiveConfiguration` parameters, see the Alchemer Mobile Configuration optional parameters section.

```
class MyApplication : Application() {
  override fun onCreate() {
    super.onCreate()
    val configuration = ApptentiveConfiguration(
    apptentiveKey = "<YOUR_APPTENTIVE_KEY>",
    apptentiveSignature = "<YOUR_APPTENTIVE_SIGNATURE>"
    ).apply {
      /**
      * Optional parameters:
      * shouldInheritAppTheme - Default is true
      * logLevel - Default is LogLevel.Info
      * shouldSanitizeLogMessages - Default is true
      * ratingInteractionThrottleLength - Default is TimeUnit.DAYS.toMillis(7)
      * customAppStoreURL - Default is null (Rating Interaction attempts to show Google In-App Review)
      */
    }
    Apptentive.register(this, configuration)
  }
}
```

If you didn't already have a `Application` class defined in your app, you will need to create one and add it to your Manifest.

Register Activity

With the re-write of the SDK, we leveraged a modern architecture using modules. This allows for greater flexibility in where the `engage` method can be called from.

Now the current `Activity` needs to register to the SDK in order to show our Interactions in your application.

Since this will need to be done for every `Activity` within the application, we recommend that you implement this within a `BaseActivity` that your other Activities can extend from.

```
class MainActivity : AppCompatActivity(), ApptentiveActivityInfo {
  override fun onResume() {
    super.onResume()
    // Register the activity callback
    Apptentive.registerApptentiveActivityInfoCallback(this)
  }

  // Pass the current Activity
  override fun getApptentiveActivityInfo(): Activity {
    return this
  }

  override fun onPause() {
    // (Optional)Unregister the activity callback
    Apptentive.unregisterApptentiveActivityInfoCallback(this)
    super.onPause()
  }
}
```

Styling Alchemer Mobile

Alchemer Mobile will inherit your app's styles by default. If you want to customize more, check this article on our interface customization currently available and how the default interaction UIs look.

Prerequisites

- If you haven't already, you will need to update your app to use Material Components and AndroidX
  - This should be a simple process and is highly recommended.
  - There are Bridge themes available if you cannot inherit them from the MaterialComponents theme.
- If you **CANNOT** use a Material theme, you can add all the colors that we use so that your Alchemer Mobile interactions are still styled correctly.
  - At the very least, `colorSurface` and `colorOnSurface` are required to avoid errors.
  - Adding `colorError` is also recommended to avoid the non-material default of an orange error color.

Engage Events

Events record user interactions. You can use them to determine if and when to show an interaction 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.

`Events` can be added *almost** anywhere in the App. A few good places would be when an `Activity` comes to focus, on a button tap or when the App encounters an error.

> * Avoid calling engage events in your `Application` class or before the SDK has a chance to get fully registered.

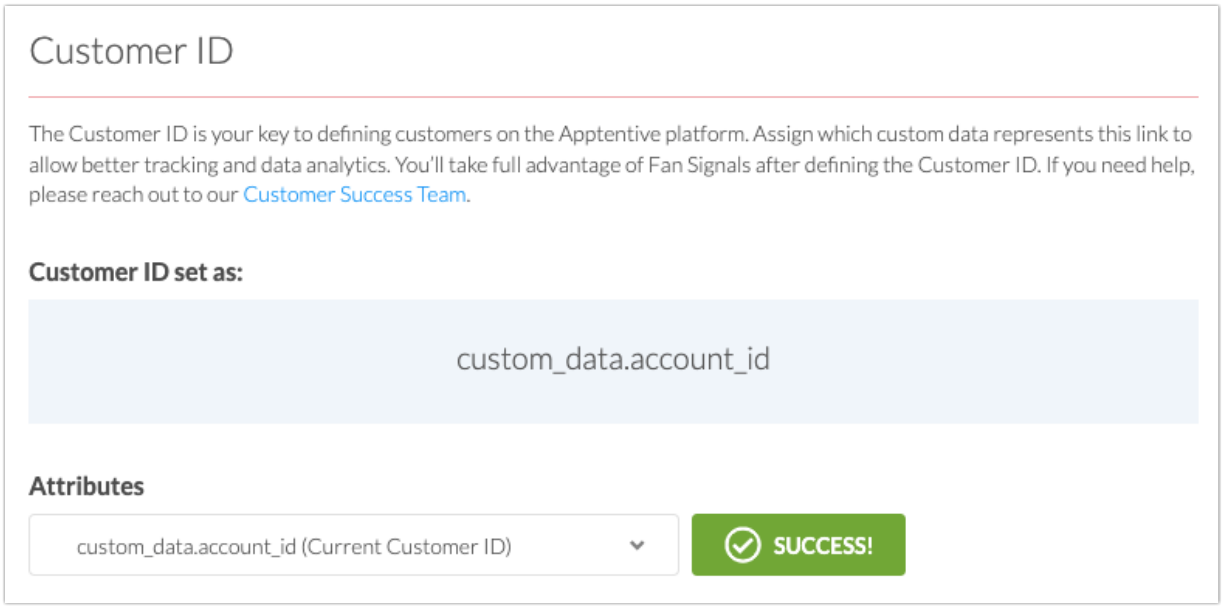
```
// Engaging
Apptentive.engage("my_event")

// Engaging with callback (optional)
Apptentive.engage("my_event") { result ->
    when (result) {
        is EngagementResult.InteractionShown -> { /* Interaction was shown */ }
        is EngagementResult.InteractionNotShown -> { /* Interaction was NOT shown */ }
        is EngagementResult.Error -> { /* There was an error during evaluation */ }
        is EngagementResult.Exception -> { /* Something went wrong */ }
    }
}
```

Add Custom Data

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 be used for configuring when Interactions will run. You can add custom data of type `String`, `Number`, and `Boolean`.

```
Apptentive.addCustomPersonData("customer_id", 1234567890)
Apptentive.addCustomPersonData(“is_premium”, true)
```

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



If you are interested in learning more about the Customer ID feature, please review this article.

Additionally, the customer’s name and email can be set by using the following APIs.

```
Apptentive.setPersonName("John Doe")
Apptentive.setPersonEmail("customer@abc.com")
```

Create Interactions

Now that you’ve created an Event, let’s see how to create Interactions and display them when the `Event` is triggered.

In this example, we will see how to create a Survey and display it.

- Go to the Surveys page.
- Click “New Survey”.
- Give the Survey a name, title, introduction, add a question, choose whether to end with a *Thank You* message, and finally click **Save & Continue**.
- Choose *Publish survey as an independent Interaction*.
- Under the **Where** section, choose the Event `main_activity_created` (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.
- Near the bottom, check *Allow multiple responses from the same person* so you can display

this survey more than once.

- Click **Save & Continue.**
- Click **Launch Survey.**
- Finally, **uninstall and 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_activity_created` Event.

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 `Button` that will launch it when clicked. This example assumes you have an `Activity` called `SettingsActivity` on that, you have a `Button` that you would like to open Message Center with.

```
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.settings_layout)
  val button = findViewById<Button>(R.id.message_center_button)
  button.setOnClickListener {
    Apptentive.showMessageCenter()
  }
  Apptentive.canShowMessageCenter { showMessageCenter ->
    button.isVisible = showMessageCenter
  }
}
```

Example App

Check out our Example App to see how Alchemer Mobile is integrated into it. Follow the README.md for the setup.

> `MainActivity.kt` shows how to engage all the interactions and `styles.xml` has the configuration for all the cookbook designs.

Related Articles