Alchemer Mobile mParticle Integration Guide: Android

This document will guide you through setting up the Alchemer Mobile (Apptentive) mParticle Kit through your existing mParticle integration. If you are not using mParticle, you should use the native integration documentation.

See our system requirements for the kit version < 5+. New System requirements for the kit version starting from 5.50.4

Check out our example app.

View GitHub details here and the build.gradle here.

1. Adding Alchemer Mobile (formerly Apptentive) to Your App

In your `build.gradle`, add the Alchemer Mobile (formerly Apptentive) mParticle kit alongside the mParticle core dependency

> Find the latest version on maven central

```
implementation 'com.mparticle:android-apptentive-kit:5+'
implementation 'com.apptentive:apptentive-kit-android:6.0.3'
```

Register Activity

Starting from mParticle kit version 5.50.4 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.

Kotlin:

```
class MainActivity : AppCompatActivity(), ApptentiveActivityInfo {
    override fun onResume() {
        super.onResume()
        ApptentiveKitUtils.registerApptentiveActivityContext(this)
    }

    override fun getApptentiveActivityInfo(): Activity {
        return this
    }
}
```

Java:

```
public class MainActivity extends AppCompatActivity implements ApptentiveActivityInfo {
   @Override
   protected void onResume() {
      super.onResume();
      ApptentiveKitUtils.registerApptentiveActivityContext(this);
   }

   @NonNull
   @Override
   public Activity getApptentiveActivityInfo() {
      return this;
   }
}
```

2. Set Up Alchemer Mobile (formerly Apptentive) in the mParticle Dashboard

Add Output: Alchemer Mobile (Apptentive)

You'll need an Alchemer Mobile Dashboard to proceed. Need help creating one? Reach out to CSM or reach out to Alchemer Support. Let us know the name of your company and apps, then we'll get you set up as soon as possible.

1. Make sure you have access to a new Android app Dashboard on Alchemer Mobile. If you need a new Android Dashboard, click here to create a new one .
2. In the mParticle Directory, click *Apptentive*
3. Click *+Add Apptentive to Setup*
4. Check the box **Output: Event**
5. Click *Add to Setup*
6. In the next screen, give your configuration the name **Android**
7. Using the values found in Alchemer Mobile's API & Development page of your Alchemer Mobile Dashbaord, fill in the Alchemer Mobile **API Key** and **Alchemer Mobile API Signature** fields in mParticle
8. Click **Save**

Connect Android Input to the Apptentive Output

1. In the mParticle menu, click *Connections*
2. Under **Available Inputs**, click *Android*
3. Under **Connected Outputs**, click *Connect Output*

4. Click *Apptentive*
5. Select Configuration *Android*
6. Turn **Status** on to *Active*
7. Click *Add Connection*

> Note
>
> There may be a delay of several minutes before this configuration is downloaded by mParticle, and Alchemer Mobile is enabled in your app.

3. Configure Events

If you have already been using mParticle in your app, you will most likely have calls to `MParticle.getInstance().logEvent()`. These calls will be delegated through to Alchemer Mobile (Apptentive), and the Event names you passed into `logEvent()` will be available for segmentation and targeting of Alchemer Mobile (Apptentive) Interactions. If you haven't already logged Events in your app, you should do so now.

We recommend sending 20 to 50 Events to Alchemer Mobile (Apptentive), and these can be adjusted at any time. Instead of sending *all* Events, focus on those that you'd like to use for segmentation (e.g. show a Survey to customers that have triggered x Event) or for displaying Alchemer Mobile dialogs (e.g. show a Love Dialog on x page when that Event is triggered).

Once Events have been toggled on in your mParticle dashboard, then triggered within your apps, they will be displayed on your Alchemer Mobile Events page.

There are two kinds of events that are able to be passed from mParticle to Alchemer Mobile (Apptentive). Using mParticle's terminology, those are: "Events" and "Screens". Details on each are below.

mParticle "Events"

- Naming conventions:
  - "Events" in mParticle = "Events" in Apptentive
  - "Event Attributes" in mParticle = "Event Custom Data" in Apptentive

"Event Attributes", which are nested under the events themselves, cannot be used in Alchemer Mobile (Apptentive).

mParticle "Screens"

- Naming convention:
  - "Screens" or "screen events" in mParticle = Events in Alchemer Mobile (Apptentive)

Description from mParticle: Screen events are a special event type used for tracking navigation within your app... For most use cases, the best course of action will be to log your navigation

events as screen views and let mParticle translate your data into the appropriate format for each output integration. Many output integrations are only interested in the screen name, *but you can also include a set of custom attributes with a screen event.*

Uncheck the box to "Send new data points by default." Like any other Events, you should send through only screen events that will be *useful* to use in targeting Alchemer Mobile interactions.

As described above, mParticle allows companies to define **screen event attributes.** Similar to Event Attributes, Alchemer Mobile does not support this type of data for targeting at this time.

4. Configure Custom Data

User Attributes in mParticle can be sent to Alchemer Mobile (Apptentive) and used as Person Custom Data. Alchemer Mobile (Apptentive) allows you to use this data in two ways.

First, it can be used in targeting Alchemer Mobile Interactions. For example, you could show an Alchemer Mobile Love Dialog only to customers with a specific Person Custom Data value.

Second, you can see these values in many types of Alchemer Mobile reporting. For example, if you displayed a Survey to all customers, you could see the Person Custom Data values for specific customers in line with their responses to the Survey.

If you have already been using User Attributes on mParticle, you can send them to Apptentive as Custom Data from the Connections page of mParticle. **We recommend sending 5 to 10 Person Custom Data fields to Alchemer Mobile (Apptentive), focusing on those that would be most useful to you. Please do not send all fields or any PII.**

Create an instance of the current user to add custom data to:

```
val currentUser = MParticle.getInstance().Identity().currentUser
```

Set the user's key/value attributes using user.setUserAttribute(). You can add String, Integer, or Boolean values as user attributes to send as Person Custom Data, but keep in mind all values are converted to Strings:

```
currentUser.setUserAttribute("location_region", "Iceland")
currentUser.setUserAttribute("number_books_read", 45)
currentUser.setUserAttribute("have_gone_to_italy", true)
```

If you want to remove an attribute from a user:

```
currentUser.removeUserAttribute("top_region");
```

Once sent through, Alchemer Mobile will automatically detect whether each user attribute is a boolean, string, or number. User attributes that are detected as either a boolean or number will be sent twice – once as a string and once with a suffix added to show the detected type "flag" (boolean) or "number". The different types will allow you enhanced targeting options, such as "greater than" and "less than" targeting for integers.

5. Interactions

The Alchemer Mobile (Apptentive) kit for mParticle allows you to use every kind of Alchemer Mobile Interaction. All you need to do is configure them in your Alchemer Mobile Dashboard via the Interactions tab. Details on each type can be found below.

Love Dialog & Rating Dialog

Love Dialogs can help learn about your customers, ask customers that love your app to rate it in the applicable app store, and ask customers who don't love it yet to give you feedback or answer a Survey.

Prompting customers to leave ratings and reviews with an in-app Rating Dialog is a great way to engage customers and request feedback.

> See: How to Use the Love Dialog and Rating Dialog

Surveys

Surveys are a powerful tool for learning about your customers' needs.

> See: How to Use Surveys

Prompts

Prompts (formerly Notes) allow you to show an alert to customers, and optionally direct them to a Survey, Message Center, Deep Link, or simply dismiss the Prompt.

> See: How to Use Prompts

Message Center

> See: How to Use Message Center

A vital part of our product is the ability to talk to your customers using Message Center. The mParticle API doesn't have a concept of feedback or messaging, so you will need to call into our native SDK directly.

```
Apptentive.showMessageCenter()
```

You should find a place in your app where you can add a feedback button that will open Message Center. Because Alchemer Mobile (Apptentive) may not be enabled via your mParticle dashboard, and because you may have disabled Message Center in the Alchemer Mobile dashboard, you

should check for availability before you show your feedback button. Here is an example:

Kotlin:

```
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_main)
  val button: Button = findViewById(R.id.feedback_button)

  //Check whether Message Center is available
  val messageCenterEnabled =
    Mpartile.getInstance()?.isKitActive(MParticle.ServiceProviders.APPTENTIVE)
    == true && Apptentive.canShowMessageCenter()

  // Show or hide the feedback button
  feedbackButton.isVisible = messageCenterEnabled

  // Open Message Center when the feedback button is clicked
  feedbackButton.setOnClickListener {
     Apptentive.showMessageCenter()
  }
}
```

Java:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Button feedbackButton = findViewById(R.id.feedback_button);

 // Check whether Message Center is available
 boolean messageCenterEnabled =
  MParticle.getInstance().isProviderActive(ServiceProviders.APPTENTIVE)
  && Apptentive.canShowMessageCenter();

 // Show or hide the feedback button
 if (messageCenterEnabled) {
  feedbackButton.setVisibility(View.VISIBLE);
 } else {
  feedbackButton.setVisibility(View.GONE);
 }

 // Open Message Center when the feedback button is clicked
 feedbackButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
   Apptentive.showMessageCenter();
  }
 });
}
```

6. Other Alchemer Mobile Features

Customizing the Look and Feel

By default, Alchemer Mobile Surveys and Message Center will inherit your global styling. If you'd like, you can override many of those global styles.

For full details, refer to our Interface Customization Guide.

Please note that Love Dialogs, Prompts, and Alchemer Mobile Rating Dialogs use default OS alert styling. The Google Play Rating Dialog uses styling provided by Google.

How to Use a Specific Apptentive SDK Version

We do not use dynamic versioning for our Android SDKs. If the latest release version is not automatically resolved when using the mParticle build, you can explicitly specify the desired version in your project.
To do so, update your app/build.gradle file as follows:

```
implementation ('com.mparticle:android-apptentive-kit:5+') {
   exclude group: 'com.apptentive', module: 'apptentive-kit-android'}
implementation 'com.apptentive:apptentive-kit-android:<sdk-version>'
```

By following this approach, you ensure your app uses the exact Apptentive SDK version you need, regardless of the version included in the mParticle dependency.

Related Articles