mParticle Migration Guide

This is for customers who move from our native SDK integration to the mParticle Kit integration.

Technical Migration:

1. It's important to reach out to your Customer Success Manager or contact support if you are planning to migrate over so we can work with you on timeline and next steps.

2. Add the mParticle Alchemer Mobile (Apptentive) Kit to your dependencies:

Android:

```
dependencies {
  implementation 'com.mparticle:android-apptentive-kit:<KIT_VERSION>'
}
```

iOS:

```
pod 'mParticle-Apptentive'
```

1. Remove the SDK:
   - Remove the entirety of the Alchemer Mobile (Apptentive) configuration
   - Remove the Alchemer Mobile (Apptentive) SDK from app dependencies, it is still present as part of the Alchemer Mobile (Apptentive) mP kit

2. Add Message Center (if being used):
   - If you want to start using a *standalone (launched from a native button)* Message Center via the mP kit, you will need to add it: Android | iOS
   - If you are already using a *standalone* Message Center via the native SDK, you don't need to make any changes to the code that launches it. However, you should add a check to see if the Alchemer Mobile (Apptentive) service is active through mP:

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

1. Make sure you have Events configured properly:
   - You will need to convert all Apptentive.engage() calls to mP event builds (see bottom). If you do not do this, both sets of events will appear in your Apptentive dashboard.
   - If it's a net new mP integration, you can use the same event names as you did with the native SDK, if you wish. This could help make it a cleaner experience and require fewer updates to your interaction targeting.
   - If you have already integrated mP, let your CSM know. We will review to ensure you have

good events available and they are being passed through. We will also work with you to ensure your targeting is properly updated.
    - Note that if you made new event names you will need to migrate all existing targeting to their new mParticle events and clone any surveys running with native event targeting.
        - You may also want to archive old events on your dashboard once targeting is updated to avoid confusion from the new/old integrations.

2. Custom Data
    - You will need to convert all Apptentive.setCustomPersonData() / Apptentive.setCustomDeviceData() to setting user attributes using mP (see bottom).
    - We do not support Device Custom data, if you want to send that data you will need to set it as user attribute / person custom data.

3. Please ensure that you toggle the events and custom data within mParticle and don't default to sending ALL:
    - We do not support event custom data or "event attributes" as it's called in mP, however we do support custom data, screen views, and events.

4. Confirm with your CSM what your timeline is for integration and launch with the Alchemer Mobile (Apptentive) kit

Code Comparisons between Native and mParticle integrations

To engage events in mParticle, change Apptentive Engage() calls to building an event with the event name, then calling logEvent() using the mParticle instance and passing the built event.

For Custom Data, change addCustomerPersonData() calls to get the instance of the current user, then set the attribute with the custom data key/value.

ANDROID NATIVE APPTENTIVE SDK: (Java and Kotlin)

```
 // Engage the Event 'video_watched'
Apptentive.engage(this, "video_watched");

// Add Custom Data
Apptentive.addCustomPersonData("location_region", "Iceland");
Apptentive.addCustomDeviceData("camera_available", true);
```

ANDROID APPTENTIVE SDK THROUGH mPARTICLE:

```
 // Engage the event 'video_watched'
val event = MPEvent.Builder("video_watched").build();
MParticle.getInstance().logEvent(event);

// Add Custom Data (User Attributes);
val currentUser = MParticle.getInstance()!!.Identity().currentUser;
currentUser.setUserAttribute("location_region", "Iceland");
currentUser.setUserAttribute("camera_available", "true");
```

iOS NATIVE APPTENTIVE SDK (Objective-C and Swift)

```
 // Engage the Event 'video_watched'
Apptentive.shared.engage(event: "video_watched", from: self);

// Add Custom Data
Apptentive.shared.addCustomPersonData("Iceland", withKey: "location_region");
Apptentive.shared.addCustomDeviceData(true, withKey: "camera_available");
```

iOS APPTENTIVE SDK THROUGH mPARTICLE: (Objective-C)

```
 // Engage the Event 'video_watched'
MPEvent *event = [[MPEvent alloc] initWithName:@"video_watched"
                                          type:MPEventTypeOther];
[[MParticle sharedInstance] logEvent:event];

// Add Custom Data
MParticleUser *currentUser = [[[MParticle sharedInstance] identity] currentUser];
[currentUser setUserAttribute:@"location_region"
                        value:@"Iceland"];
[currentUser setUserAttribute:@"camera_available"
                        value:@"true"];
```

iOS APPTENTIVE SDK THROUGH mPARTICLE: (Swift)

```
 // Engage the Event 'video_watched'
if let event = MPEvent(name: "video_watched", type: MPEventType.other) {
    MParticle.sharedInstance().logEvent(event)
}

// Add Custom Data
let currentUser = MParticle.sharedInstance().identity.currentUser
currentUser?.setUserAttribute("location_region", value: "Europe")
currentUser?.setUserAttribute("camera_available", value: "true")
```

Related Articles