Alchemer Mobile Cordova SDK Plugin Guide

This document will show you how to integrate and configure the Alchemer Mobile Cordova SDK Plugin into your Cordova app.

Cordova Versions

Required Minimum Versions

Starting with Cordova 6.2.0, we require at least a minimum version of the plugins below:

Cordova Platform: **10.0.1**

Cordova Android: **9.0.0**

Cordova iOS: **4.3.1**

Gradle version: >= 7.0.0

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

Supported Platforms

This Cordova Plugin wraps our existing native Android and iOS SDKs. The complete descriptions for the features contained in our SDKs are located here:

- Android API level 21+
- iOS version 11+

Registering Apps

You will need to create an app in your Alchemer Mobile account for each of the platforms you wish to use Alchemer Mobile on. Even though there is a single plugin for Cordova that works on both Android and iOS, you will need to create separate Android and iOS apps on our website.

Once you have created Android and iOS apps, you will need to access the Alchemer Mobile App Key and Alchemer Mobile App Signature for each app here.

Adding the Plugin to Your App

Our plugin is hosted on NPMJS or GitHub. Installation is easy and involves pointing at our repo and passing in both of the API Keys you got from the previous step.

```
cordova plugin add apptentive-cordova --variable ANDROID_APP_KEY="YOUR_ANDROID_KEY" --variable ANDROID_APP_SIGNATURE="YOUR_ANDROID_SIGNATURE" --variable IOS_APP_KEY="YOUR_IOS_KEY" --variable IOS_APP_SIGNATURE="YOUR_IOS_SIGNATURE"
```

Do not reuse the same Alchemer Mobile Keys for Android and iOS.

If you have trouble upgrading to a new version of Alchemer Mobile, try removing the plugin and re-adding it. In rare cases, Cordova may be in a bad state, and you may have to remove all plugins and platforms, then re-add them.

Support Material Dialog Colors

To successfully support Android Material dialog colors on Cordova you can add this code block to your `config.xml` file. This will simply add the `colorSurface` and `colorOnSurface` style items to your app's theme that our dialog interactions rely on.

```
<?xml version='1.0' encoding='utf-8'?>
<widget ...>
  <platform name="android">
    <config-file target="res/values/themes.xml" parent="/resources">
      <style name="ApptentiveThemeOverride">
        <item name="colorSurface">?android:colorBackground</item>
        <item name="colorOnSurface">?colorOnBackground</item>
      </style>
    </config-file>
  </platform>
</widget>
```

Optional Configuration Options

Most apps will only need to set their KEY and SIGNATURE values. The rest of these values are all **optional**.

Configuration name	Description	Default value (if not set)

Configuration name	Description	Default value (if not set)
`ANDROID_LOG_LEVEL`	An `enum` used to define what level of logs we will show in Android Studio's Logcat. All log levels (in order of most verbose to least) is `verbose`, `debug`, `info`, `warning`, and `error`. When debugging or sending logs to Alchemer Mobile, please change your log level to `verbose` to capture as much info as possible.	`info`
`ANDROID_USES_DEVICE_ENCRYPTION`	Encrypts Alchemer Mobile SDK info on the device. If your application maintain sensitive user data (account numbers, health information, etc) and you pass it to Alchemer Mobile SDK (as a custom person/device data) – you may want to enable encrypted device storage. Most apps will **not** need to use this feature.	`false`
`ANDROID_SANITIZE_LOG_MESSAGES`, `IOS_SANITIZE_LOG_MESSAGES`	Redacts certain information from being logged (see Android documentation and iOS documentation) When debugging or sending logs to our Customer Success team, please change this value to `false` to capture as much important info as possible.	`true`

Configuration name	Description	Default value (if not set)
`ANDROID_INHERIT_APP_THEME`	Determines if Alchemer Mobile Interactions will use the host app's theme or Alchemer's theme. Setting this value to `true` will inherit the native android app's theming from your `themes.xml` file.	`false`
`ANDROID_CUSTOM_APP_STORE_URL`	Used for alternate app store ratings. When this value is `null`, the consumer will be prompted with the Google In-APp Review when asked to rate the app. Setting this value to a URL will force the SDK to use the Alchemer Mobile Rating Dialog instead of Google's In-App Review and send the consumer to the specified URL when the RATE button is tapped. It is recommended to **not** set this if you are wanting consumers to rate your app that is in the Google Play Store.	`null`

Configuration name	Description	Default value (if not set)
`ANDROID_RATING_INTERACTION_THROTTLE_LENGTH`	A millisecond based rating interaction throttle. Fallback safeguard to prevent consumers being prompted to rate the app too often. Google limits the amount of times their In-App Review can be shown, and this helps prevent wasting those.	`604800000` (7 days)

Example command with default values

```
cordova plugin add apptentive-cordova --variable ANDROID_APP_KEY="YOUR_ANDROID_KEY" --variable ANDROID_APP_SIGNATURE="YOUR_ANDROID_SIGNATURE" --variable IOS_APP_KEY="YOUR_IOS_KEY" --variable IOS_APP_SIGNATURE="YOUR_IOS_SIGNATURE" --variable ANDROID_LOG_LEVEL=info --variable ANDROID_INHERIT_APP_THEME=true --variable ANDROID_USES_DEVICE_ENCRYPTION=false --variable ANDROID_SANITIZE_LOG_MESSAGES=true --variable IOS_SANITIZE_LOG_MESSAGES=true --variable ANDROID_CUSTOM_APP_STORE_URL=" " --variable ANDROID_RATING_INTERACTION_THROTTLE_LENGTH="604800000"
```

> While the default of `ANDROID_CUSTOM_APP_STORE_URL` is set to a blank string above (our Cordova default), do **not** set it if you wish to use Google's In-App Review. Only set it if you wish to use Alchemer Mobile's Rating Dialog Interaction.

Using Alchemer Mobile 6.2.0+

> Please pay attention to this section

The native Android library was updated with a major version with release 6.2.0 and a couple changes are needed to integrate successfully.

Android SDK Level

We have updated our minimum supported `cordova` and `cordova-android` versions to 10.0.1 and 9.0.0.

Background:
Every new Android version introduces changes that enhance the user experience, security, and performance of the Android platform overall. Each app specifies a `targetSdkVersion` (also known as

the target API level) in the manifest file. The target API level indicates how your app is meant to run on different Android versions.

Google Play periodically raises the minimum SDK-level requirements in order to upload an app to the Play Store. As of November 2022 the minimum required version is **Android 12 (API level 31)**. After August 2023, the minimum required version will be **Android 13 (API level 33)**.

More info on Google's SDK-level requirements here.

How to upgrade:

Cordova's latest `cordova-android` versions target a high enough SDK level (API level 31) to submit to the Play Store, but be sure to stay up to date on those.

Info on migrating to `cordova-android 11` here.

> *If you are having issues, please reach out to your CSM or contact Alchemer Support, and our support team will help get you up-and-running.*

Initialize Alchemer Mobile

Make sure you call `Apptentive.deviceReady()` when your app has started.

```
onDeviceReady: function() {
 Apptentive.deviceReady(successCallback, failureCallback);
}
```

Or if you want to debug your apptentive integration, call `Apptentive.registerWithLog()` when your app has started. Use "verbose" to get all logs from Apptentive, or "info" to get the default logs.

```
onDeviceReady: function() {
 Apptentive.registerWithLogs(successCallback, failureCallback, "verbose");
}
```

> Do not use both `deviceReady` and `registerWithLogs`.
>
> They do the same thing except `registerWithLogs` lets you specify the log level.

Adding Events

You should add a handful of **Events** to your app when you integrate. Since **Events** are both records of an action within your app being performed, and an opportunity to show an **Interaction**, you should choose places within your app that would be appropriate to interact with your customer, as well as places where a significant event has occured. The more Events you add during integration, the more you will learn about your customers, and the more fine-tuned your communications with

them can be. Here is a list of potential places to add **Events**.

Places where you might want to show an **Interaction**:

- The app opens
- Settings view gains focus
- The consumer performs an action that indicates they might need help
- There is a natural pause in the app's UI where starting a conversation would not interrupt the customer

Places where you might want to record an **Event**:

- The consumer makes a purchase
- The consumer declines to make a purchase
- The consumer performs an action that indicates they know how to use your app
- The consumer performs an action that indicates they need help

As you can see, there is some overlap in whether you want to just record an **Event**, or also show an **Interaction**.

To add an **Event** and possibly show an **Interaction**, simply call `Apptentive.engage(successCallback, errorCallback, eventName)` with an `eventName` of your choosing.

```
Apptentive.engage(successCallback, failureCallback, 'eventName');
```

> Each **Event** should have a unique name.

Interactions

Once you have configured your app to use several **Events**, you can configure **Interactions** on apptentive.com

First make sure you have created a few Events in your app.

Love Dialog

Go to *Interactions -> Love Dialog*. In the tab labeled *The Dialog*, you can customize the text and behavior of the dialog that makes up the **Love Dialog** interaction. The *Targeting* tab lets you create user segments that will see the **Love Dialog**, as well as define the conditions necessary for the **Love Dialog** to display. You will also need to pick which **Event** will trigger the **Love Dialog** to be shown, by entering an **Event** name in the *Where* section of that page.

Prompts

Prompts (formerly Notes) are configured from the dashboard. Go to *Interactions -> Prompts*. Create a new Prompts. You can give it a title, content, and buttons. After your Prompt is live, you

will start to see results in the Prompts *-> Reporting* section.

Survey

Surveys are configured from the dashboard. Go to *Interactions -> Surveys*. Create a new survey. You can give it a title and description, then add questions, and finally set targeting and limiting constraints so it's shown to the right people. After your survey is live, you will start to see results in the *Surveys -> Reporting* section.

Survey finished listener

To know when a consumer has submitted a survey, you can add a listener.

```
Apptentive.addSurveyFinishedListener(successCallback, failureCallback);
```

Message Center

The Message Center is a messaging UI that allows you to talk with your customer. You should find a place in your app where you can create a link or button that opens your **Message Center**.

```
Apptentive.showMessageCenter(successCallback, failureCallback);
```

New Message Notification

You can be notified any time the number of unread messages in Message Center changes, by registering a listener when your device finishes initializing. You can add this next to your existing call to `Apptentive.deviceReady();`

```
// To receive the unread message count once
Apptentive.getUnreadMessageCount(successCallback, errorCallback);

// To receive updates of the unread message count
Apptentive.addUnreadMessagesListener(successCallback, errorCallback);
```

When the number of unread messages changes, this listener will be called, and an integer representing the number of unread messages will be passed to it.

```
 const successCallback = () => {
  console.log('Apptentive device ready');
};

const failureCallback = () => {
  console.error('Apptentive device not ready');
}

const unreadMessagesListenerSuccess = (count) => {
  console.log("UnreadMessages: " + count);
};

const unreadMessagesListenerFailure = (errorMessage) => {
  console.log("Error: " + errorMessage);
};

document.addEventListener('deviceready', () => {
    Apptentive.registerWithLogs(successCallback, failureCallback, "verbose");

    Apptentive.addUnreadMessagesListener(unreadMessagesListenerSuccess, unreadMessagesListenerFailure)
;
});
```

Can Show Interaction / Message Center

To determine if sending an event through the engage function will show an Interaction, or if the showMessageCenter function will show Message Center, use the functions below.

```
 Apptentive.canShowInteraction(canShowInteractionSuccess, canShowInteractionFailure, eventName);

Apptentive.canShowMessageCenter(canShowMessageCenterSuccess, canShowMessageCenterFailure);
```

Set Customer Contact Information

If you already know the customer's email address, you can pass it to us during initialization. Simply call

```
 Apptentive.setPersonEmail(successCallback, errorCallback, email);
```

If you know their name, and would like to see it displayed when you are communicating with them through the Alchemer Mobile dashboard, call

```
 Apptentive.setPersonName(successCallback, errorCallback, name);
```

Custom Data

You can send Custom Data associated with either the device, or the person using the app. This is useful for sending user IDs and other information that helps you support your users better. **Custom Data** can also be used for configuring when Interactions will run.

Supported Custom Data types are `String`, `Number`, and `Boolean`

```
Apptentive.addCustomDeviceData(successCallback, errorCallback, key, value);
```

```
Apptentive.addCustomPersonData(successCallback, errorCallback, key, value);
```

Hidden Message Center messages

> Starting from Alchemer Mobile Cordova 6.1.1

If there is additional info you'd like to send to the server that will help with working with a consumer in Message Center, you can send hidden messages will not be shown on the device, but will be shown in support agent's message thread with the consumer.

```
Apptentive.sendAttachmentText(successCallback, errorCallback, `Additional info for support agent: ${info}.`
);
```

Unsupported Features

If you are interested in any of the below features, please let us know at Alchemer Support.

Multi User

Multi-user is not currently supported by our Cordova Plugin. **This is actively being worked on and should be implemented soon.**

Push Notifications

Push notifications are not currently supported by our Cordova Plugin.

Hidden Attachments

Sending hidden attachments is not currently supported by our Cordova Plugin.

Related Articles