Summary

This is the Unity SDK of Adjust™. It supports iOS, Android, Windows Store 8.1, Windows Phone 8.1 and Windows 10 targets. You can read more about Adjust™ at adjust.com.

Note: As of version **4.12.0**, Adjust Unity SDK is compatible with **Unity 5 and newer** versions.

Note: As of version **4.19.2**, Adjust Unity SDK is compatible with **Unity 2017.1.1 and newer** versions.

Note: As of version **4.21.0**, Adjust Unity SDK is compatible with **Unity 2017.4.1 and newer** versions.

Read this in other languages: English, 中文, 日本語, 한국어.

Table of contents

Quick start

Deeplinking

Event tracking

Custom parameters

Additional features

Testing and troubleshooting

License

Quick start

Getting started

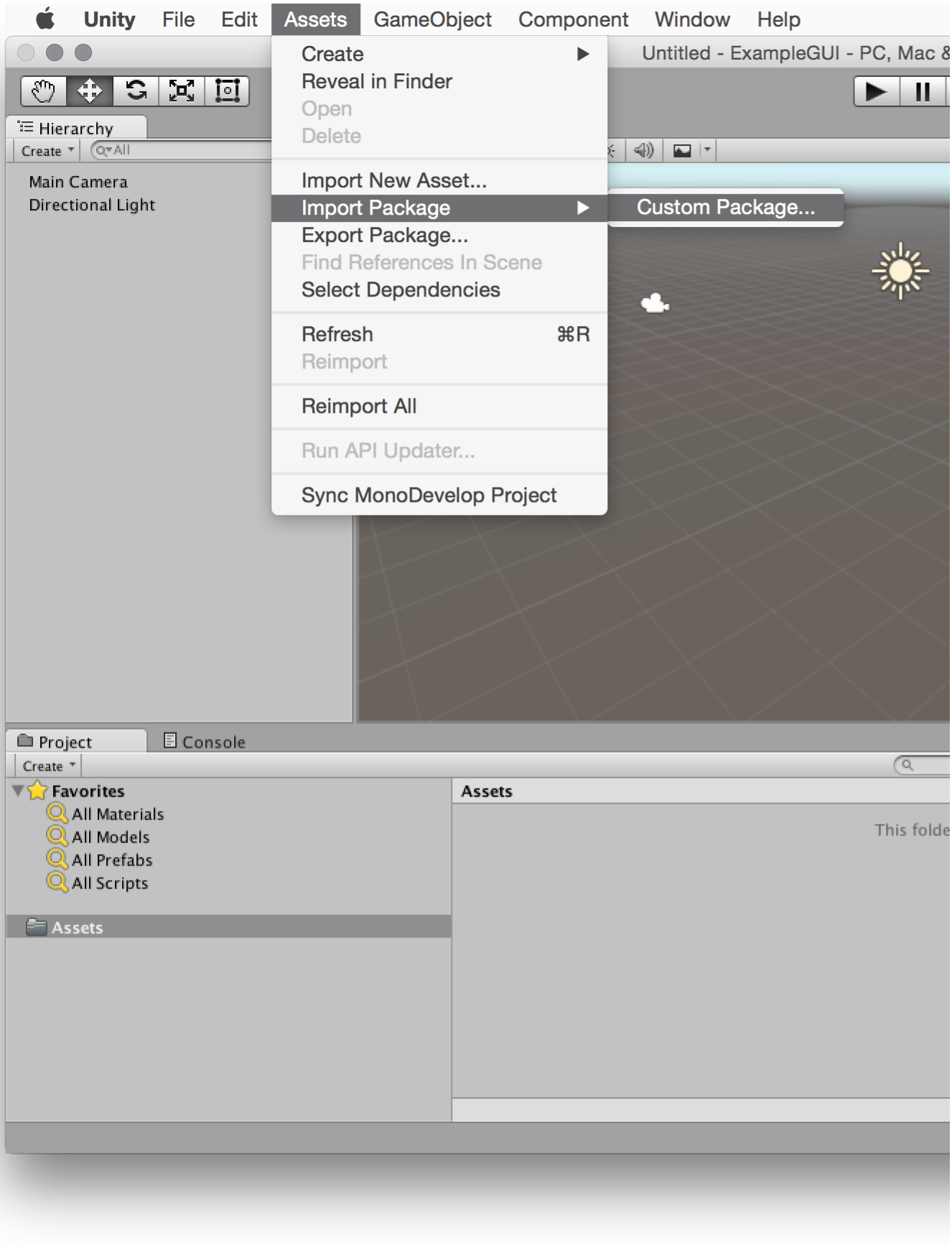
To integrate the Adjust SDK into your Unity project, follow these steps.

Get the SDK

As of version `4.19.2`, you can add Adjust SDK from Unity Asset Store to your app. Alternativly, you can download the latest version from our releases page.

Add the SDK to your project

Open your project in the Unity Editor, go to `Assets → Import Package → Custom Package` and select the downloaded Unity package file.

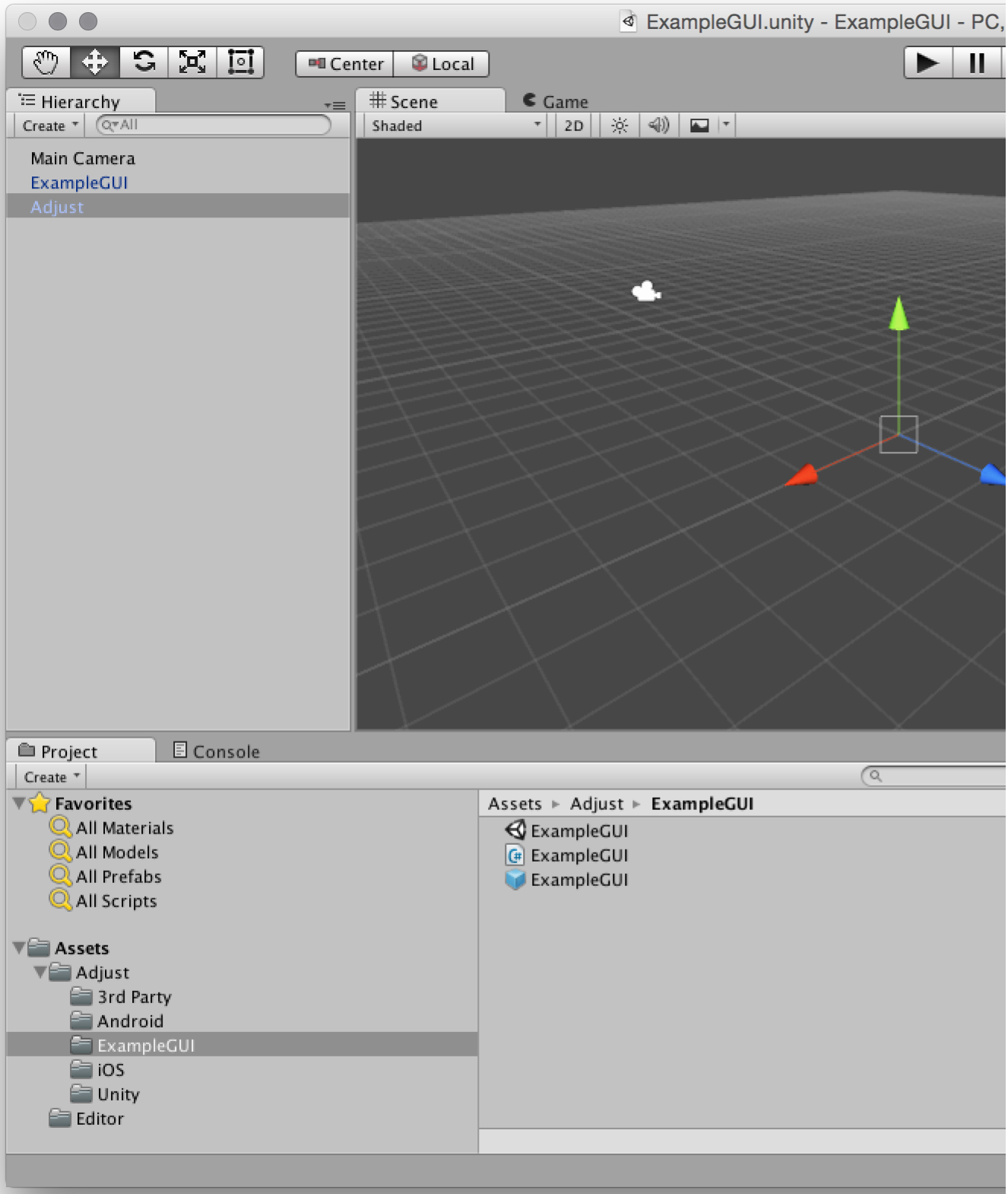


Integrate the SDK into your app

Add the prefab from `Assets/Adjust/Adjust.prefab` to the first scene.

You can edit the Adjust script parameters in the prefab `Inspector menu` to set up the following options:

- Start Manually
- Event Buffering
- Send In Background
- Launch Deferred Deeplink
- App Token
- Log Level
- Environment



Replace `{YourAppToken}` with your actual App Token. Follow these steps to find it in the dashboard.

Depending on whether you are building your app for testing or for production, change the `Environment` setting to either 'Sandbox' or 'Production'.

Important: Set the value to `Sandbox` if you or someone else is testing your app. Make sure to set the environment to `Production` before you publish the app. Set it back to `Sandbox` if you start testing again. Also, have in mind that by default Adjust dashboard is showing production traffic of your app, so in case you want to see traffic you generated while testing in sandbox mode, make sure to switch to sandbox traffic view within dashboard.

We use the environment setting to distinguish between real traffic and artificial traffic from test devices. Please make sure to keep your environment setting updated.

If you don't want the Adjust SDK to start automatically with the app's `Awake` event, select `Start Manually`. With this option, you'll initialize and start the Adjust SDK from the within the code by calling the `Adjust.start` method with the `AdjustConfig` object as a parameter.

You can find an example scene with a button menu showing these options here: `Assets/Adjust/ExampleGUI/ExampleGUI.unity`.

The source for this scene is located at `Assets/Adjust/ExampleGUI/ExampleGUI.cs`.

Adjust logging

You can increase or decrease the granularity of the logs you see by changing the value of `Log Level` to one of the following:

- `Verbose` - enable all logs
- `Debug` - disable verbose logs
- `Info` - disable debug logs (default)
- `Warn` - disable info logs
- `Error` - disable warning logs
- `Assert` - disable error logs
- `Suppress` - disable all logs

If you want to disable all of your log output when initializing the Adjust SDK manually, set the log level to suppress and use a constructor for the `AdjustConfig` object. This opens a boolean parameter where you can enter whether the suppress log level should be supported or not:

```
string appToken = "{YourAppToken}";
AdjustEnvironment environment = AdjustEnvironment.Sandbox;

AdjustConfig config = new AdjustConfig(appToken, environment, true);
config.setLogLevel(AdjustLogLevel.Suppress);

Adjust.start(config);
```

If your target is Windows-based and you want to see the compiled logs from our library in `released` mode, redirect the log output to your app while testing it in `debug` mode.

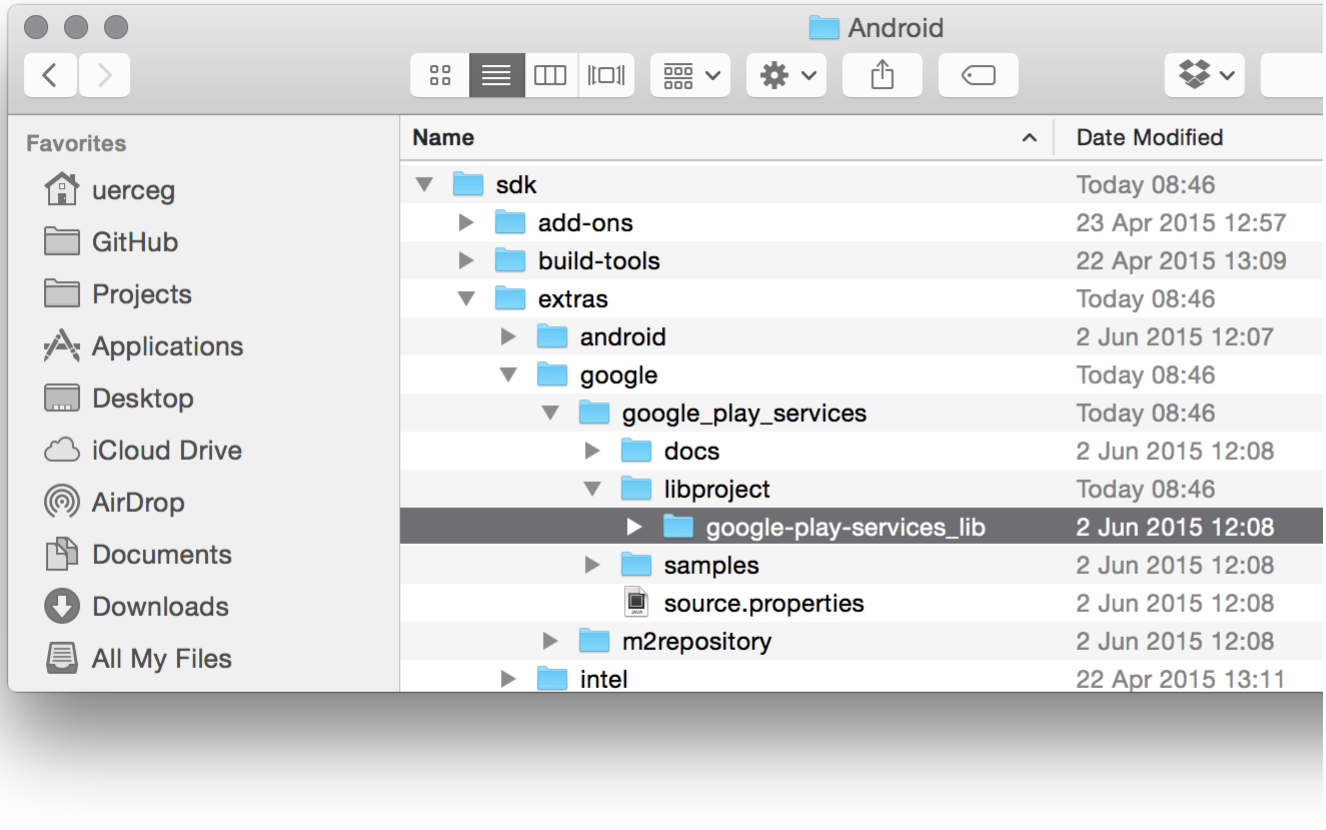
Call the method `setLogDelegate` in the `AdjustConfig` instance before starting the SDK.

```
//...
adjustConfig.setLogDelegate(msg => Debug.Log(msg));
//...
Adjust.start(adjustConfig);
```

Google Play Services

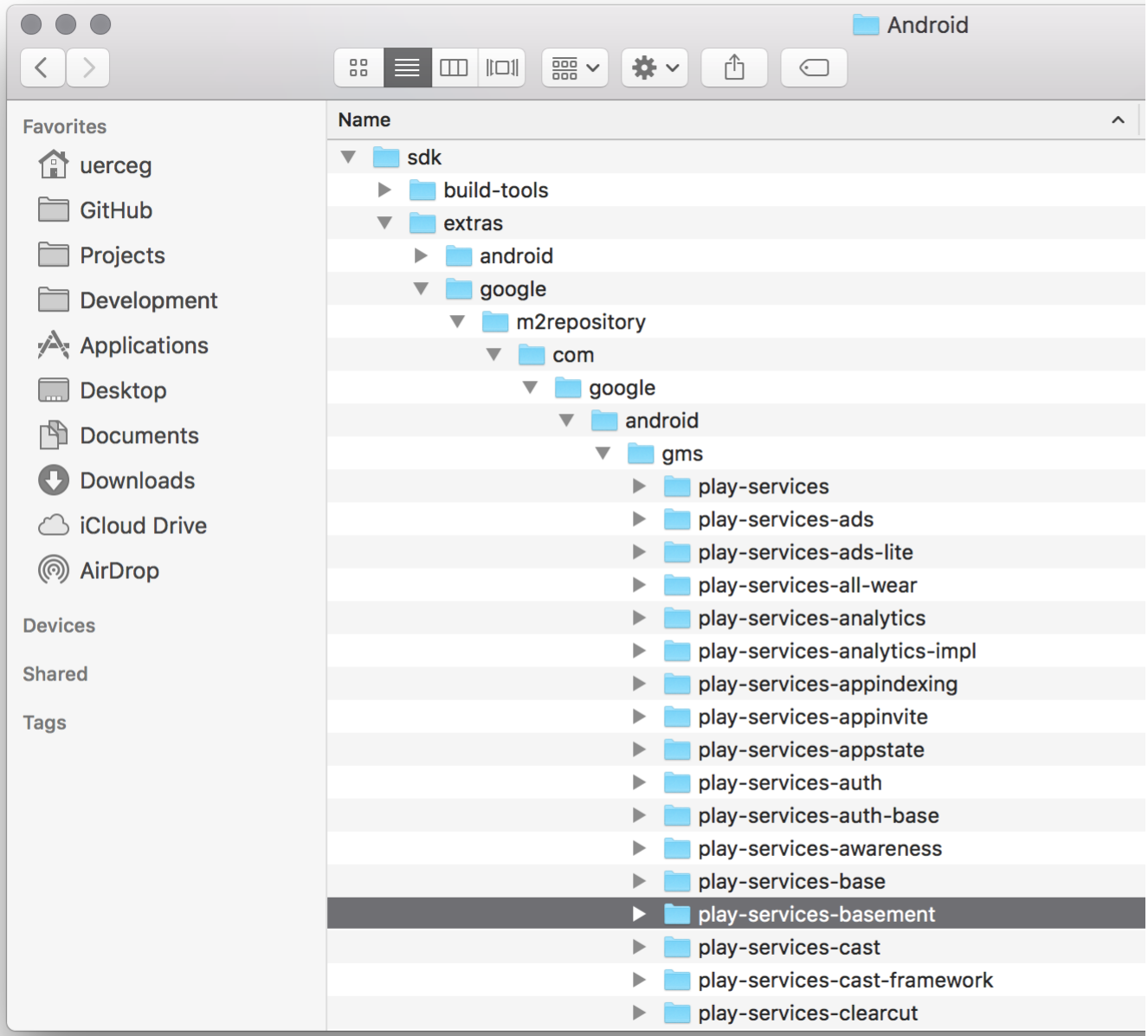
Since August 1st 2014, apps in the Google Play Store must use the Google Advertising ID to uniquely identify devices. To allow the Adjust SDK to use the Google Advertising ID, integrate Google Play Services. To do this, copy the `google-play-services_lib` folder (part of the Android SDK) into the `Assets/Plugins/Android` folder of your Unity project.

There are two main ways to download the Android SDK. Any tool using the `Android SDK Manager` will offer a quick link to downlaod and install the Android SDK tools. Once installed, you can find the libraries in the `SDK_FOLDER/extras/google/google_play_services/libproject/` folder.



If you aren't using any tools with the Android SDK Manager, download the official standalone Android SDK. Next, download the Andoird SDK Tools by following the instructions in the `SDK Readme.txt` README provided by Google, located in the Android SDK folder.

Update: With the latest Android SDK version, Google has changed the structure of the Google Play Services folders inside of the root SDK folder. It now looks like this:



You can now add only the part of the Google Play Services library that the Adjust SDK needs—the basement. To do this, add the `play-services-basement-x.y.z.aar` file to your `Assets/Plugins/Android` folder.

With Google Play Services library 15.0.0, Google has moved the classes needed to get the Google advertising ID into a `play-services-ads-identifier` package. Add this package to your app if you are using library version 15.0.0 or later. When you're finished, please test to make sure the Adjust SDK correctly obtains the Google advertising ID; we have noticed some inconsistencies, depending upon which Unity integrated development environment (IDE) version you use.

Testing for the Google advertising ID

To check whether the Adjust SDK is receiving the Google advertising ID, start your app by configuring the SDK to run in `sandbox` mode and set the log level to `verbose`. After that, track a session or an event in the app and check the list of parameters recorded in the verbose logs. If you see the `gps_adid` parameter, our SDK has successfully read the Google advertising ID.

If you encounter any issues getting the Google advertising ID, please open an issue in our Github repository or contact support@adjust.com.

Proguard settings

If you are using Proguard, add these lines to your Proguard file:

```
-keep public class com.adjust.sdk.** { *; }
-keep class com.google.android.gms.common.ConnectionResult {
    int SUCCESS;
}
-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient {
    com.google.android.gms.ads.identifier.AdvertisingIdClient$Info getAdvertisingIdInfo(android.content.Context);
}
-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient$Info {
    java.lang.String getId();
    boolean isLimitAdTrackingEnabled();
}
-keep public class com.android.installreferrer.** { *; }
```

Google Install Referrer

In order to attribute the install of an Android app, Adjust needs information about the Google install referrer. You can set up your app to get this by using the **Google Play Referrer API** or by catching the **Google Play Store intent** with a broadcast receiver.

Google introduced the Google Play Referrer API in order to provide a more reliable and secure way than the Google Play Store intent to obtain install referrer information and to help attribution providers fight click injections. The Google Play Store intent will exist in parallel with the API temporarily, but is set to be deprecated in the future. We encourage you to support this.

The Adjust post-build process catches the Google Play Store intent; you can take a few additional steps to add support for the new Google Play Referrer API.

To add support for the Google Play Referrer API, download the install referrer library from Maven repository and place the AAR file into your `Plugins/Android` folder.

Huawei Referrer API

As of v4.21.1, the Adjust SDK supports install tracking on Huawei devices with Huawei App Gallery version 10.4 and higher. No additional integration steps are needed to start using the Huawei Referrer API.

Post-build process

To complete the app build process, the Adjust Unity package performs custom post-build actions to ensure the Adjust SDK can work properly inside the app.

This process is performed by the `OnPostprocessBuild` method in `AdjustEditor.cs`. Log output messages are written to the Unity IDE console output window.

iOS post-build process

To execute the iOS post-build process properly, use Unity 5 or later and have `iOS build support` installed. The iOS post-build process makes the following changes to your generated Xcode project:

- Adds the `iAd.framework` (needed for Apple Search Ads tracking)
- Adds the `AdSupport.framework` (needed for reading IDFA)
- Adds the `CoreTelephony.framework` (needed for reading type of network device is connected to)
- Adds the other linker flag `-ObjC` (needed to recognize Adjust Objective-C categories during build time)
- Enables `Objective-C exceptions`

In case you enable iOS 14 support (`Assets/Adjust/Toggle iOS 14 Support`), iOS post-build process will add two additional frameworks to your Xcode project:

- Adds the `AppTrackingTransparency.framework` (needed to ask for user's consent to be tracked and obtain status of that consent)
- Adds the `StoreKit.framework` (needed for communication with SKAdNetwork framework)

Android post-build process

The Android post-build process makes changes to the `AndroidManifest.xml` file located in `Assets/Plugins/Android/`. It also checks for the presence of the `AndroidManifest.xml` file in the Android plugins folder. If the file is not there, it creates a copy from our compatible manifest file `AdjustAndroidManifest.xml`. If there is already an `AndroidManifest.xml` file, it makes the following changes:

- Adds the `INTERNET` permission (needed for Internet connection)
- Adds the `ACCESS_WIFI_STATE` permission (needed if you are not distributing your app via the Play Store)
- Adds the `ACCESS_NETWORK_STATE` permission (needed for reading type of network device is connected to)
- Adds the `BIND_GET_INSTALL_REFERRER_SERVICE` permission (needed for the new Google install referrer API to work)
- Adds the Adjust broadcast receiver (needed for getting install referrer information via Google Play Store intent). For more details, consult the official Android SDK README.

Note: If you are using your own broadcast receiver to handle the `INSTALL_REFERRER` intent, you don't need to add the Adjust broadcast receiver to your manifest file. Remove it, but add the call to the Adjust broadcast receiver inside your own receiver, as described in the Android guide.

SDK signature

An account manager can activate the Adjust SDK signature for you. Contact Adjust support at support@adjust.com if you want to use this feature.

If the SDK signature is enabled on your account and you have access to App Secrets in your dashboard, add all secret parameters (`secretId`, `info1`, `info2`, `info3`, `info4`) to the `setAppSecret` method of `AdjustConfig` instance:

```
AdjustConfig adjustConfig = new AdjustConfig("{YourAppToken}", "{YourEnvironment}");

adjustConfig.setAppSecret(secretId, info1, info2, info3, info4);

Adjust.start(adjustConfig);
```

The SDK signature is now integrated in your app.

Deeplinking

Deeplinking Overview

We support deeplinking on iOS and Android platforms.

If you are using Adjust tracker URLs with deeplinking enabled, it is possible to receive information about the deeplink URL and its content. Users may interact with the URL regardless of whether they have your app installed on their device (standard deeplinking) or not (deferred deeplinking).

With standard deeplinking, the Android platform lets you receive deeplink content; however, Android does not automatically support deferred deeplinking. To access deferred deeplink content, you can use the Adjust SDK.

Set up deeplink handling in your app on a **native level** within your generated Xcode project (for iOS) and Android Studio / Eclipse project (for Android).

Standard deeplinking

Information about standard deeplinks cannot be delivered to you in Unity C# code. Once you enable your app to handle deeplinking, you'll get information about the deeplink on a native level. For more information, here's how to enable deeplinking for Android and iOS apps.

Deferred deeplinking

In order to get content information about the deferred deeplink, set a callback method on the `AdjustConfig` object. This will receive one `string` parameter where the content of the URL is delivered. Set this method on the config object by calling the method `setDeferredDeeplinkDelegate`:

```
// ...

private void DeferredDeeplinkCallback(string deeplinkURL) {
   Debug.Log("Deeplink URL: " + deeplinkURL);

   // ...
}

AdjustConfig adjustConfig = new AdjustConfig("{YourAppToken}", "{YourEnvironment}");

adjustConfig.setDeferredDeeplinkDelegate(DeferredDeeplinkCallback);

Adjust.start(adjustConfig);
```

With deferred deeplinking, there is an additional setting you can set on the `AdjustConfig` object. Once the Adjust SDK gets the deferred deeplink information, you can choose whether our SDK should open the URL or not. You can set this option by calling the `setLaunchDeferredDeeplink` method on the config object:

```
// ...

private void DeferredDeeplinkCallback(string deeplinkURL) {
   Debug.Log ("Deeplink URL: " + deeplinkURL);

   // ...
}

AdjustConfig adjustConfig = new AdjustConfig("{YourAppToken}", "{YourEnvironment}");

adjustConfig.setLaunchDeferredDeeplink(true);
adjustConfig.setDeferredDeeplinkDelegate(DeferredDeeplinkCallback);

Adjust.start(adjustConfig);
```

If nothing is set, **the Adjust SDK will always try to launch the URL by default**.

To enable your apps to support deeplinking, set up schemes for each supported platform.

Deeplink handling in Android apps

To set up deeplink handling in an Android app on a native level, follow the instructions in our official Android SDK README.

This should be done in native Android Studio / Eclipse project.

Deeplink handling in iOS apps

This should be done in native Xcode project.

To set up deeplink handling in an iOS app on a nativel level, please use a native Xcode project and follow the instructions in our official iOS SDK README.

Event tracking

Track an event

You can use Adjust to track any event in your app. If you want to track every tap on a button, create a new event token in your dashboard. Let's say that the event token is `abc123`. In your button's click handler method, add the following lines to track the click:

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");
Adjust.trackEvent(adjustEvent);
```

Track revenue

If your users generate revenue by engaging with advertisements or making in-app purchases, you can track this with events. For example: if one add tap is worth one Euro cent, you can track the revenue event like this:

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");
adjustEvent.setRevenue(0.01, "EUR");
Adjust.trackEvent(adjustEvent);
```

When you set a currency token, Adjust will automatically convert the incoming revenues using the openexchange API into a reporting revenue of your choice. Read more about currency conversion here.

If you want to track in-app purchases, please make sure to call `trackEvent` only if the purchase is finished and the item has been purchased. This is important in order to avoid tracking revenue your users did not actually generate.

Revenue deduplication

Add an optional transaction ID to avoid tracking duplicated revenues. The SDK remembers the last ten transaction IDs and skips revenue events with duplicate transaction IDs. This is especially useful for tracking in-app purchases.

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");

adjustEvent.setRevenue(0.01, "EUR");
adjustEvent.setTransactionId("transactionId");

Adjust.trackEvent(adjustEvent);
```

In-app purchase verification

Verify in-app purchases using Adjust's Purchase Verification, a server-side receipt verification tool.

Custom parameters

Custom parameters overview

In addition to the data points the Adjust SDK collects by default, you can use the Adjust SDK to track and add as many custom values as you need (user IDs, product IDs, etc.) to the event or session. Custom parameters are only available as raw data and will **not** appear in your Adjust dashboard.

Use callback parameters for the values you collect for your own internal use, and partner parameters for those you share with external partners. If a value (e.g. product ID) is tracked both for internal use and external partner use, we recommend using both callback and partner parameters.

Event parameters

Event callback parameters

If you register a callback URL for events in your dashboard, we will send a GET request to that URL whenever the event is tracked. You can also put key-value pairs in an object and pass it to the `trackEvent` method. We will then append these parameters to your callback URL.

For example, if you've registered the URL `http://www.example.com/callback`, then you would track an event like this:

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");

adjustEvent.addCallbackParameter("key", "value");
adjustEvent.addCallbackParameter("foo", "bar");

Adjust.trackEvent(adjustEvent);
```

In this case we would track the event and send a request to:

```
http://www.example.com/callback?key=value&foo=bar
```

Adjust supports a variety of placeholders, for example `{idfa}` for iOS or `{gps_adid}` for Android, which can be used as parameter values. Using this example, in the resulting callback we would replace the placeholder with the IDFA/ Google Play Services ID of the current device. Read more about real-time callbacks and see our full list of placeholders.

Note: We don't store any of your custom parameters. We only append them to your callbacks. If you haven't registered a callback for an event, we will not read these parameters.

Event partner parameters

Once your parameters are activated in the dashboard, you can send them to your network partners. Read more about module partners and their extended integration.

This works the same way as callback parameters; add them by calling the `addPartnerParameter` method on your `AdjustEvent` instance.

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");

adjustEvent.addPartnerParameter("key", "value");
adjustEvent.addPartnerParameter("foo", "bar");

Adjust.trackEvent(adjustEvent);
```

You can read more about special partners and these integrations in our guide to special partners.

Event callback identifier

You can add custom string identifiers to each event you want to track. We report this identifier in your event callbacks, letting you know which event was successfully tracked. Set the identifier by calling the `setCallbackId` method on your `AdjustEvent` instance:

```
AdjustEvent adjustEvent = new AdjustEvent("abc123");

adjustEvent.setCallbackId("Your-Custom-Id");

Adjust.trackEvent(adjustEvent);
```

Session parameters

Session parameters are saved locally and sent with every Adjust SDK **event and session**. Whenever you add these parameters, we save them (so you don't need to add them again). Adding the same parameter twice will have no effect.

It's possible to send session parameters before the Adjust SDK has launched. Using the SDK delay, you can therefore retrieve additional values (for instance, an authentication token from the app's server), so that all information can be sent at once with the SDK's initialization.

Session callback parameters

You can save event callback parameters to be sent with every Adjust SDK session.

The session callback parameters' interface is similar to the one for event callback parameters. Instead of adding the key and its value to an event, add them via a call to the `addSessionCallbackParameter` method of the `Adjust` instance:

```
Adjust.addSessionCallbackParameter("foo", "bar");
```

Session callback parameters merge with event callback parameters, sending all of the information as one, but event callback parameters take precedence over session callback parameters. If you add an event callback parameter with the same key as a session callback parameter, we will show the event value.

You can remove a specific session callback parameter by passing the desired key to the `removeSessionCallbackParameter` method of the `Adjust` instance.

```
Adjust.removeSessionCallbackParameter("foo");
```

To remove all keys and their corresponding values from the session callback parameters, you can reset them with the `resetSessionCallbackParameters` method of the `Adjust` instance.

```
Adjust.resetSessionCallbackParameters();
```

Session partner parameters

In the same way that session callback parameters are sent with every event or session that triggers our SDK, there are also session partner parameters.

These are transmitted to network partners for all of the integrations activated in your dashboard.

The session partner parameters interface is similar to the event partner parameters interface, however instead of adding the key and its value to an event, add it by calling the `addSessionPartnerParameter` method of the `Adjust` instance.

```
Adjust.addSessionPartnerParameter("foo", "bar");
```

Session partner parameters merge with event partner parameters. However, event partner parameters take precedence over session partner parameters. If you add an event partner parameter with the same key as a session partner parameter, we will show the event value.

To remove a specific session partner parameter, pass the desired key to the `removeSessionPartnerParameter` method of the `Adjust` instance.

```
Adjust.removeSessionPartnerParameter("foo");
```

To remove all keys and their corresponding values from the session partner parameters, reset it with the `resetSessionPartnerParameters` method of the `Adjust` instance.

```
Adjust.resetSessionPartnerParameters();
```

Delay start

Delaying the start of the Adjust SDK gives your app time to receive any session parameters (such as unique identifiers) you may want to send on install.

Set the initial delay time in seconds with the method `setDelayStart` in the `AdjustConfig` instance:

```
adjustConfig.setDelayStart(5.5);
```

In this example, the Adjust SDK is prevented from sending the initial install session and any new event for 5.5 seconds. After 5.5 seconds (or if you call `Adjust.sendFirstPackages()` during that time), every session parameter is added to the delayed install session and events, and the Adjust SDK will work as usual.

You can delay the start time of the Adjust SDK for a maximum of 10 seconds.

Additional features

Once you integrate the Adjust SDK into your project, you can take advantage of the following features:

AppTrackingTransparency framework

Note: This feature exists only in iOS platform.

For each package sent, the Adjust backend receives one of the following four (4) states of consent for access to app-related data that can be used for tracking the user or the device:

- Authorized
- Denied
- Not Determined
- Restricted

After a device receives an authorization request to approve access to app-related data, which is used for user device tracking, the returned status will either be Authorized or Denied.

Before a device receives an authorization request for access to app-related data, which is used for tracking the user or device, the returned status will be Not Determined.

If authorization to use app tracking data is restricted, the returned status will be Restricted.

The SDK has a built-in mechanism to receive an updated status after a user responds to the pop-up dialog, in case you don't want to customize your displayed dialog pop-up. To conveniently and efficiently communicate the new state of consent to the backend, Adjust SDK offers a wrapper around the app tracking authorization method described in the following chapter, App-tracking authorization wrapper.

App-tracking authorisation wrapper

Note: This feature exists only in iOS platform.

Adjust SDK offers the possibility to use it for requesting user authorization in accessing their app-related data. Adjust SDK has a wrapper built on top of the requestTrackingAuthorizationWithCompletionHandler: method, where you can as well define the callback method to get information about a user's choice. Also, with the use of this wrapper, as soon as a user responds to the pop-up dialog, it's then communicated back using your callback method. The SDK will also inform the backend of the user's choice. The `NSUInteger` value will be delivered via your callback method with the following meaning:

- 0: `ATTrackingManagerAuthorizationStatusNotDetermined`
- 1: `ATTrackingManagerAuthorizationStatusRestricted`
- 2: `ATTrackingManagerAuthorizationStatusDenied`
- 3: `ATTrackingManagerAuthorizationStatusAuthorized`

To use this wrapper, you can call it as such:

```
Adjust.requestTrackingAuthorizationWithCompletionHandler((status) =>
{
    switch (status)
    {
        case 0:
            // ATTrackingManagerAuthorizationStatusNotDetermined case
            break;
        case 1:
            // ATTrackingManagerAuthorizationStatusRestricted case
            break;
        case 2:
            // ATTrackingManagerAuthorizationStatusDenied case
            break;
        case 3:
            // ATTrackingManagerAuthorizationStatusAuthorized case
            break;
    }
});
```

SKAdNetwork framework

Note: This feature exists only in iOS platform.

If you have implemented the Adjust iOS SDK v4.23.0 or above and your app is running on iOS 14, the communication with SKAdNetwork will be set on by default, although you can choose to turn it off. When set on, Adjust automatically registers for SKAdNetwork attribution when the SDK is initialized. If events are set up in the Adjust dashboard to receive conversion values, the Adjust backend sends the conversion value data to the SDK. The SDK then sets the conversion value. After Adjust receives the SKAdNetwork callback data, it is then displayed in the dashboard.

In case you don't want the Adjust SDK to automatically communicate with SKAdNetwork, you can disable that by calling the following method on configuration object:

```
adjustConfig.deactivateSKAdNetworkHandling();
```

Push token (uninstall tracking)

Push tokens are used for Audience Builder and client callbacks; they are also required for uninstall and reinstall tracking.

To send us a push notification token, call the `setDeviceToken` method on the `Adjust` instance when you obtain your app's push notification token (or whenever its value changes):

```
Adjust.setDeviceToken("YourPushNotificationToken");
```

Attribution callback

You can set up a callback to be notified about attribution changes. We consider a variety of different sources for attribution, so we provide this information asynchronously. Make sure to consider applicable attribution data policies before sharing any of your data with third-parties.

Follow these steps to add the optional callback in your application:

1. Create a method with the signature of the delegate `Action<AdjustAttribution>`.
2. After creating the `AdjustConfig` object, call the `adjustConfig.setAttributionChangedDelegate` with the previously created method. You can also use a lambda with the same signature.
3. If instead of using the `Adjust.prefab` the `Adjust.cs` script was added to another `GameObject`, be sure to pass the name of the `GameObject` as the second parameter of `AdjustConfig.setAttributionChangedDelegate`.

Because the callback is configured using the `AdjustConfig` instance, call `adjustConfig.setAttributionChangedDelegate` before calling `Adjust.start`.

```
using com.adjust.sdk;

public class ExampleGUI : MonoBehaviour {
    void OnGUI() {
        if (GUI.Button(new Rect(0, 0, Screen.width, Screen.height), "callback")) {
            AdjustConfig adjustConfig = new AdjustConfig("{Your App Token}", AdjustEnvironment.Sandbox);
            adjustConfig.setLogLevel(AdjustLogLevel.Verbose);
            adjustConfig.setAttributionChangedDelegate(this.attributionChangedDelegate);

            Adjust.start(adjustConfig);
        }
    }

    public void attributionChangedDelegate(AdjustAttribution attribution) {
        Debug.Log("Attribution changed");

        // ...
    }
}
```

The callback function will be called when the SDK receives final attribution data. Within the callback function you have access to the `attribution` parameter. Here is a quick summary of its properties:

- `string trackerToken` the tracker token of the current attribution
- `string trackerName` the tracker name of the current attribution
- `string network` the network grouping level of the current attribution
- `string campaign` the campaign grouping level of the current attribution
- `string adgroup` the ad group grouping level of the current attribution
- `string creative` the creative grouping level of the current attribution
- `string clickLabel` the click label of the current attribution
- `string adid` the Adjust device identifier

Ad revenue tracking

You can track ad revenue information with the Adjust SDK by using the following method:

```
Adjust.trackAdRevenue(source, payload);
```

The method parameters you need to pass are:

- `source` - `string` object which indicates the source of ad revenue info.
- `payload` - `string` object which contains ad revenue JSON in string form.

Currently we support the following `source` parameter values:

- `AdjustConfig.AdjustAdRevenueSourceMopub` - represents the MoPub mediation platform

Subscription tracking

Note: This feature is only available in the SDK v4.22.0 and above.

You can track App Store and Play Store subscriptions and verify their validity with the Adjust SDK. After a subscription has been successfully purchased, make the following call to the Adjust SDK:

For App Store subscription:

```
AdjustAppStoreSubscription subscription = new AdjustAppStoreSubscription(
    price,
    currency,
    transactionId,
    receipt);
subscription.setTransactionDate(transactionDate);
subscription.setSalesRegion(salesRegion);

Adjust.trackAppStoreSubscription(subscription);
```

For Play Store subscription:

```
AdjustPlayStoreSubscription subscription = new AdjustPlayStoreSubscription(
    price,
    currency,
    sku,
    orderId,
    signature,
    purchaseToken);
subscription.setPurchaseTime(purchaseTime);

Adjust.trackPlayStoreSubscription(subscription);
```

Subscription tracking parameters for App Store subscription:

- price
- currency (you need to pass currencyCode of the priceLocale object)
- transactionId
- receipt
- transactionDate
- salesRegion (you need to pass countryCode of the priceLocale object)

Subscription tracking parameters for Play Store subscription:

- price
- currency
- sku
- orderId
- signature
- purchaseToken
- purchaseTime

Note: Subscription tracking API offered by Adjust SDK expects all parameters to be passed as `string` values. Parameters described above are the ones which API exects you to pass to subscription object prior to tracking subscription. There are various libraries which are handling in app purchases in Unity and each one of them should return information described above in some form upon successfully completed subscription purchase. You should locate where these parameters are placed in response you are getting from library you are using for in app purchases, extract those values and pass them to Adjust API as string values.

Just like with event tracking, you can attach callback and partner parameters to the subscription object as well:

For App Store subscription:

```
AdjustAppStoreSubscription subscription = new AdjustAppStoreSubscription(
    price,
    currency,
    transactionId,
    receipt);
subscription.setTransactionDate(transactionDate);
subscription.setSalesRegion(salesRegion);

// add callback parameters
subscription.addCallbackParameter("key", "value");
subscription.addCallbackParameter("foo", "bar");

// add partner parameters
subscription.addPartnerParameter("key", "value");
subscription.addPartnerParameter("foo", "bar");

Adjust.trackAppStoreSubscription(subscription);
```

For Play Store subscription:

```
AdjustPlayStoreSubscription subscription = new AdjustPlayStoreSubscription(
    price,
    currency,
    sku,
    orderId,
    signature,
    purchaseToken);
subscription.setPurchaseTime(purchaseTime);

// add callback parameters
subscription.addCallbackParameter("key", "value");
subscription.addCallbackParameter("foo", "bar");

// add partner parameters
subscription.addPartnerParameter("key", "value");
subscription.addPartnerParameter("foo", "bar");

Adjust.trackPlayStoreSubscription(subscription);
```

Session and event callbacks

You can set up callbacks to notify you of successful and failed events and/or sessions.

Follow these steps to add the callback function for successfully tracked events:

```
// ...

AdjustConfig adjustConfig = new AdjustConfig("{Your App Token}", AdjustEnvironment.Sandbox);
adjustConfig.setLogLevel(AdjustLogLevel.Verbose);
adjustConfig.setEventSuccessDelegate(EventSuccessCallback);

Adjust.start(adjustConfig);

// ...

public void EventSuccessCallback(AdjustEventSuccess eventSuccessData) {
    // ...
}
```

Add the following callback function for failed tracked events:

```
// ...

AdjustConfig adjustConfig = new AdjustConfig("{Your App Token}", AdjustEnvironment.Sandbox);
adjustConfig.setLogLevel(AdjustLogLevel.Verbose);
adjustConfig.setEventFailureDelegate(EventFailureCallback);

Adjust.start(adjustConfig);

// ...

public void EventFailureCallback(AdjustEventFailure eventFailureData) {
    // ...
}
```

For successfully tracked sessions:

```
// ...

AdjustConfig adjustConfig = new AdjustConfig("{Your App Token}", AdjustEnvironment.Sandbox);
adjustConfig.setLogLevel(AdjustLogLevel.Verbose);
adjustConfig.setSessionSuccessDelegate(SessionSuccessCallback);

Adjust.start(adjustConfig);

// ...

public void SessionSuccessCallback (AdjustSessionSuccess sessionSuccessData) {
    // ...
}
```

For failed tracked sessions:

```
// ...

AdjustConfig adjustConfig = new AdjustConfig("{Your App Token}", AdjustEnvironment.Sandbox);
adjustConfig.setLogLevel(AdjustLogLevel.Verbose);
adjustConfig.setSessionFailureDelegate(SessionFailureCallback);

Adjust.start(adjustConfig);

// ...

public void SessionFailureCallback (AdjustSessionFailure sessionFailureData) {
    // ...
}
```

Callback functions will be called after the SDK tries to send a package to the server. Within the callback you have access to a response data object specifically for the callback. Here is a quick summary of the session response data properties:

- `string Message` the message from the server or the error logged by the SDK
- `string Timestamp` timestamp from the server
- `string Adid` a unique device identifier provided by Adjust
- `Dictionary<string, object> JsonResponse` the JSON object with the response from the server

Both event response data objects contain:

- `string EventToken` the event token, if the package tracked was an event
- `string CallbackId` the custom defined callback ID set on an event object

Both event and session failed objects also contain:

- `bool WillRetry` indicates there will be an attempt to resend the package at a later time

User attribution

This callback, like an attribution callback, is triggered whenever the attribution information changes. Access your user's current attribution information whenever you need it by calling the following method of the `Adjust` instance:

```
AdjustAttribution attribution = Adjust.getAttribution();
```

Note: Current attribution information is available after our backend tracks the app install and triggers the attribution callback. It is not possible to access a user's attribution value before the SDK has been initialized and the attribution callback has been triggered.

Device IDs

The Adjust SDK lets you receive device identifiers.

iOS Advertising Identifier

To obtain the IDFA, call the function `getIdfa` of the `Adjust` instance:

```
string idfa = Adjust.getIdfa();
```

Google Play Services advertising identifier

The Google advertising ID can only be read in a background thread. If you call the method `getGoogleAdId` of the `Adjust` instance with an `Action<string>` delegate, it will work in any situation:

```
Adjust.getGoogleAdId((string googleAdId) => {
    // ...
});
```

You will now have access to the Google advertising ID as the variable `googleAdId`.

Amazon advertising identifier

If you need to get the Amazon advertising ID, call the `getAmazonAdId` method on `Adjust` instance:

```
string amazonAdId = Adjust.getAmazonAdId();
```

Adjust device identifier

Our backend generates a unique Adjust device identifier (known as an `adid`) for every device that has your app installed. In order to get this identifier, call this method on `Adjust` instance:

```
String adid = Adjust.getAdid();
```

Information about the adid is only available after our backend tracks the app install. It is not possible to access the adid value before the SDK has been initialized and the installation of your app has been successfully tracked.

Pre-installed trackers

To use the Adjust SDK to recognize users whose devices came with your app pre-installed, follow these steps:

1. Create a new tracker in your dashboard.
2. Set the default tracker of your `AdjustConfig`:

```
AdjustConfig adjustConfig = new AdjustConfig(appToken, environment);
adjustConfig.setDefaultTracker("{TrackerToken}");
Adjust.start(adjustConfig);
```

   Replace `{TrackerToken}` with the tracker token you created in step 2. E.g. `{abc123}`

Although the dashboard displays a tracker URL (including `http://app.adjust.com/`), in your source code you should only enter the six or seven-character token and not the entire URL.

3. Build and run your app. You should see a line like the following in the log output:

```
Default tracker: 'abc123'
```

Offline mode

Offline mode suspends transmission to our servers while retaining tracked data to be sent at a later point. While the Adjust SDK is in offline mode, all information is saved in a file. Please be careful not to trigger too many events in offline mode.

Activate offline mode by calling `setOfflineMode` with the parameter `true`.

```
Adjust.setOfflineMode(true);
```

Deactivate offline mode by calling `setOfflineMode` with `false`. When you put the Adjust SDK back into online mode, all saved information is sent to our servers with the correct time information.

This setting is not remembered between sessions, meaning that the SDK is in online mode whenever it starts, even if the app was terminated in offline mode.

Disable tracking

You can disable Adjust SDK tracking by invoking the method `setEnabled` with the enabled parameter as `false`. This setting is remembered between sessions, but it can only be activated after the first session.

```
Adjust.setEnabled(false);
```

You can check if the Adjust SDK is currently active with the method `isEnabled`. It is always possible to activate the Adjust SDK by invoking `setEnabled` with the `enabled` parameter set to `true`.

Event buffering

If your app makes heavy use of event tracking, you might want to delay some network requests in order to send them in one batch every minute. You can enable event buffering with your `AdjustConfig` instance:

```
AdjustConfig adjustConfig = new AdjustConfig("{YourAppToken}", "{YourEnvironment}");

adjustConfig.setEventBufferingEnabled(true);

Adjust.start(adjustConfig);
```

If nothing is set, event buffering is disabled by default.

Background tracking

The default behaviour of the Adjust SDK is to pause sending network requests while the app is in the background. You can change this in your `AdjustConfig` instance:

```
AdjustConfig adjustConfig = new AdjustConfig("{YourAppToken}", "{YourEnvironment}");

adjustConfig.setSendInBackground(true);

Adjust.start(adjustConfig);
```

GDPR right to be forgotten

In accordance with article 17 of the EU's General Data Protection Regulation (GDPR), you can notify Adjust when a user has exercised their right to be forgotten. Calling the following method will instruct the Adjust SDK to communicate the user's choice to be forgotten to the Adjust backend:

```
Adjust.gdprForgetMe();
```

Upon receiving this information, Adjust will erase the user's data and the Adjust SDK will stop tracking the user. No requests from this device will be sent to Adjust in the future.

Please note that even when testing, this decision is permanent. It is not reversible.

Disable third-party sharing for specific users

You can now notify Adjust when a user has exercised their right to stop sharing their data with partners for marketing purposes, but has allowed it to be shared for statistics purposes.

Call the following method to instruct the Adjust SDK to communicate the user's choice to disable data sharing to the Adjust backend:

```
Adjust.disableThirdPartySharing();
```

Upon receiving this information, Adjust will block the sharing of that specific user's data to partners and the Adjust SDK will continue to work as usual.

Testing and troubleshooting

Debug information in iOS

Even with the post build script it is possible that the project is not ready to run out of the box.

If needed, disable dSYM File. In the `Project Navigator`, select the `Unity-iPhone` project. Click the `Build Settings` tab and search for `debug information`. There should be an `Debug Information Format` or `DEBUG_INFORMATION_FORMAT` option. Change it from `DWARF with dSYM File` to `DWARF`.

License

License agreement