SDK Integration

ByteBrew's lightwieght SDKs are compatible with every game engine and platform you use. Download our SDK's on Github

---

Unity Integration

Before you start: ByteBrew supports Android 6.0 and above & iOS version 9.0 and above. ByteBrew SDK is compatible with iOS 14 updates.

Step 1: Import ByteBrew Unity SDK to your project

Step 2: Attach the ByteBrew script to a gameobject

Create an empty gameobject and attach the ByteBrew script to the object.

Important: Place your game object on the first scene of your game. ByteBrew SDK does not support offline event caching.

Step 3: Enable Platforms

Enable which platforms (iOS & Android) you want to track on the inspector panel.

Step 4: Go to your ByteBrew Dashboard

Go to your game settings on the ByteBrew dashboard and find your game keys listed on the dashboard.

Step 5: Input game keys

Inside Unity, input the game keys from your game into the ByteBrew Unity Panel.

Step 6: Manually initialize ByteBrew

You must manually initialize ByteBrew using the code below at the first scene of your game.

```
// Initialize ByteBrew
ByteBrew.InitializeByteBrew();
```

Step 7 (optional iOS only, skip if not needed): Call ByteBrew's ATT(App Tracking Transparency) wrapper

ByteBrew offers the ability to not have to call native code in your unity application, if you are using our attribution please call this before initializing our SDK.

If you do choose to use this please make sure that you do the following in your info.plist:

1. Use the Xcode Property List Editor to add: Privacy - Tracking Usage Description

2. In the value string field enter a sentence or statement for the reason behind you tracking request.

```
// Call ByteBrew ATT Wrapper
ByteBrew.requestForAppTrackingTransparency((status) =>
{
    //Case 0: ATTrackingManagerAuthorizationStatusAuthorized
    //Case 1: ATTrackingManagerAuthorizationStatusDenied
    //Case 2: ATTrackingManagerAuthorizationStatusRestricted
    //Case 3: ATTrackingManagerAuthorizationStatusNotDetermined
    Debug.Log("ByteBrew Got a status of: " + status);
    ByteBrew.InitializeByteBrew();
});
```

Step 8: Exporting iOS Requirements

ByteBrew requires the following three iOS Frameworks:

1. Security.Framework

2. iAd.Framework

3. AdSupport.Framework

Implement these in your final xcode project as frameworks.

Tracking Purchases

To track a basic in-app purchase event utilize the below method

```
ByteBrew.TrackInAppPruchaseEvent("Apple App Store", "USD", 5.99f, "currencyPack01", "Currencies");
```

Validate Purchases

To validate and track the in-app purchase utilize the specific platform method.

Make sure the receipts have correct JSON string formatting.

```
//JSON Format ex. string json = "{\"firstname\":\"john\", \"lastname\":\"doe\",\"age\":30}";

//Retrieve the iOS receipt from the purchase event that occurs. We will validate it server side so you
string iosReciept = "...";
ByteBrew.TrackiOSInAppPruchaseEvent("Apple App Store", "USD", 5.99f, "currencyPack01", "Currencies", io

//Retrieve the Android receipt and Signature from the purchase event that occurs. We will validate it s
string googleReciept = "...";
string googleSignature = "...";
ByteBrew.TrackGoogleInAppPruchaseEvent("Google Play Store", "USD", 5.99f, "currencyPack01", "Currencies
```

Custom Event Tracking

The most basic form of custom event tracking

```
ByteBrew.NewCustomEvent("shopOpen");
```

To add more detail to an event you can add a secondary parameter, that can be a float or string value

```
//string example
ByteBrew.NewCustomEvent("weapon_equip", "megablaster");

//float example
ByteBrew.NewCustomEvent("lives_earned", 5f);
```

Progression Event Tracking

You can track a progression event multiple ways. Progression get tracked based on: ProgressionType (Started, Completed, Failed) - the type of event context that occured Environment - Area of the game or World it occurs (ex. Tutorial, Arena, Level), Stage - The stage of the environment (ex. kings_arena, jungleLevel, level_02), Value - This can be a string or float value attached to the event

```
//Example Progression event where the user started a tutorial in arena 0 or first arena
ByteBrew.NewProgressionEvent(ByteBrewProgressionTypes.Started, "Tutorial", "arena0");

//Example progression event where the user has completed the kings arena and could be rewarded with 3 c
ByteBrew.NewProgressionEvent(ByteBrewProgressionTypes.Completed, "Arena", "kings_arena", 3f);
```

Ad Event Tracking

Track ad events that occur to get more detailed breakdowns in your monetization, especially when doing LTV and ROAS calculations.

```
// Record the Placement Type, Location of the placement, ad unit ID, and Network
ByteBrew.TrackAdEvent("Interstitial", "EndOfLevel", "3253k3302-3r3j4i343-3nij343-405403", "AdMob");

//Some ad event paramters can be ommited like so
ByteBrew.TrackAdEvent("Interstitial", "EndOfLevel");
ByteBrew.TrackAdEvent("Interstitial", "EndOfLevel", "3253k3302-3r3j4i343-3nij343-405403");
```

Remote Config & AB Testing

Remote configs help you edit your apps settings and configuration without needed to update your app on the store.

AB Testing is to cross track changes across a variety of users when they onboard to your game, we distribute AB Test variables from remote configs. So make sure to check for the AB test you are tracking

When using Remote Config you first must call for the config to get updated. You can call this whenever you want to update the configs.

```
//Call the Action handler
ByteBrew.RemoteConfigsUpdated(() =>
{
    //Do Whatever you want here once the config is updated
});
```

Finally once you get the OK that the remote config has been updated call this method.

Remember AB Test values will be here as well, if the user is part of the control group it wont return anything but the default value on your retrieval line of code.

```
//Call to get the key specific value and if the key doesn't exist it will return the default variable s
ByteBrew.GetRemoteConfigForKey("dailyWeapon", "goldRevolver");
```

To view your game's remote configs visit the remote config page on the ByteBrew dashboard.