

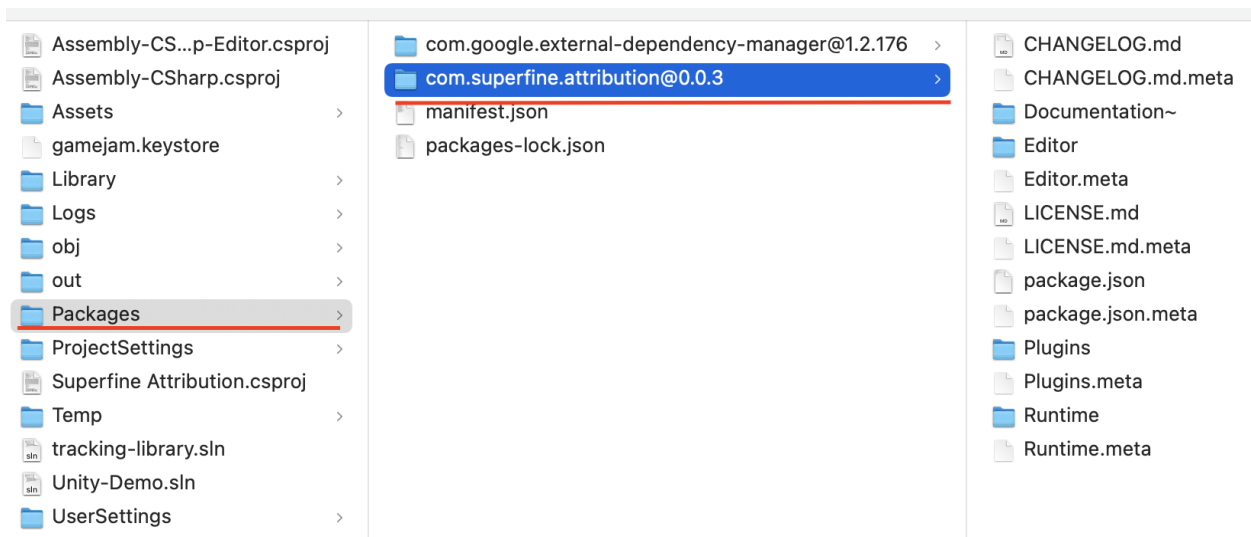
# Superfine SDK

Version 0.0.3

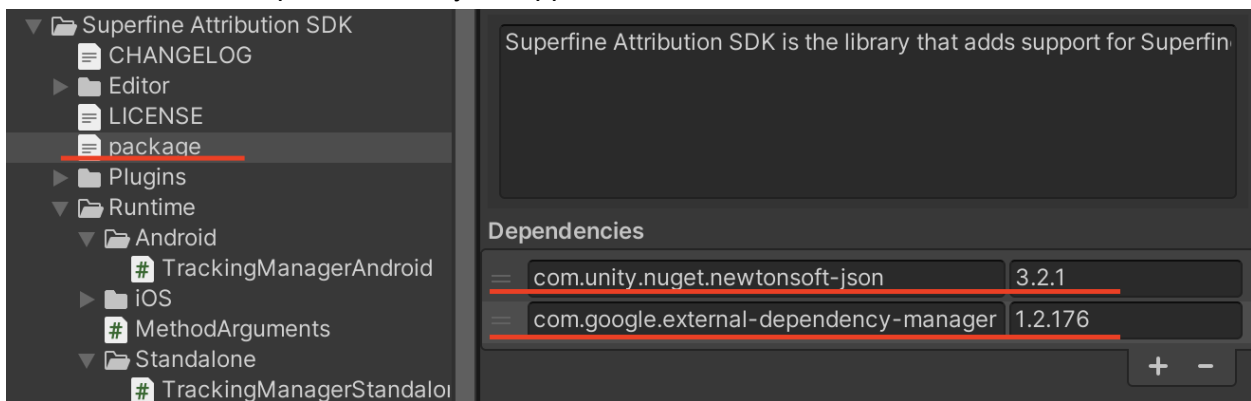
## 1 Setup

### 1.1 Import Unity package

Download the SuperfineSDK zip file, unzip it, and copy the extracted files to your Packages folder. You can find the zip file for download [HERE](#)

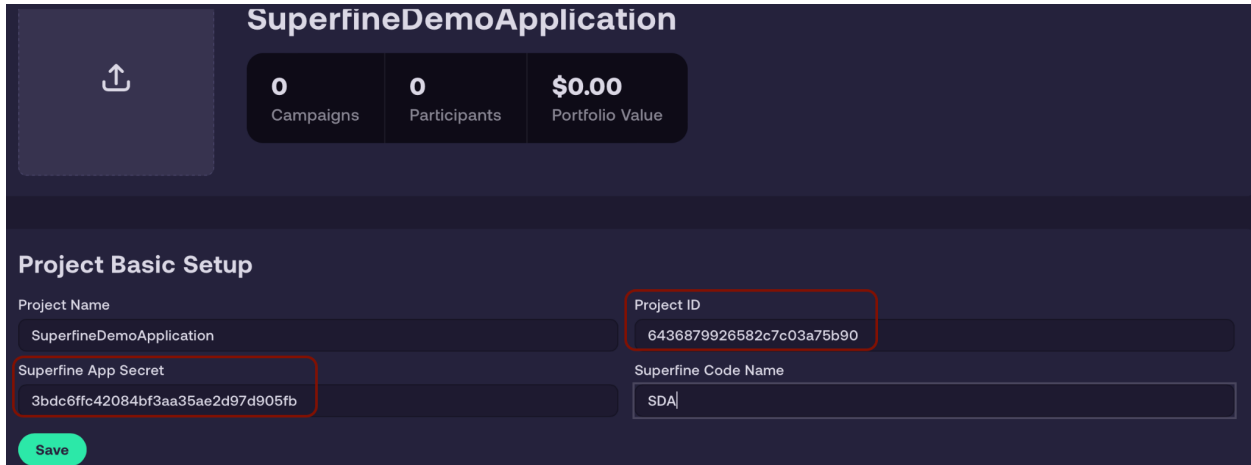


\* The SDK requires JSON.NET and external-dependency-manager to function properly. We have included them within the SDK, but you have the flexibility to remove or modify them to match the version requirements of your application.



## 1.2 Get App Information

Go to the project section on the Superfine.org dashboard, select the project, and copy the **Project ID** and **Superfine App Secret**.



**SuperfineDemoApplication**

0 Campaigns   0 Participants   \$0.00 Portfolio Value

**Project Basic Setup**

Project Name: SuperfineDemoApplication

Project ID: 6436879926582c7c03a75b90

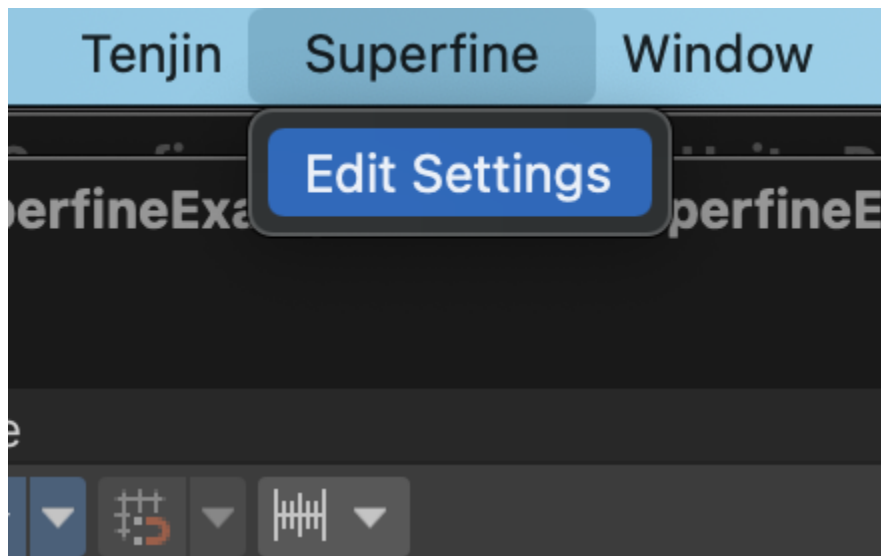
Superfine App Secret: 3bdc6ffc42084bf3aa35ae2d97d905fb

Superfine Code Name: SDA

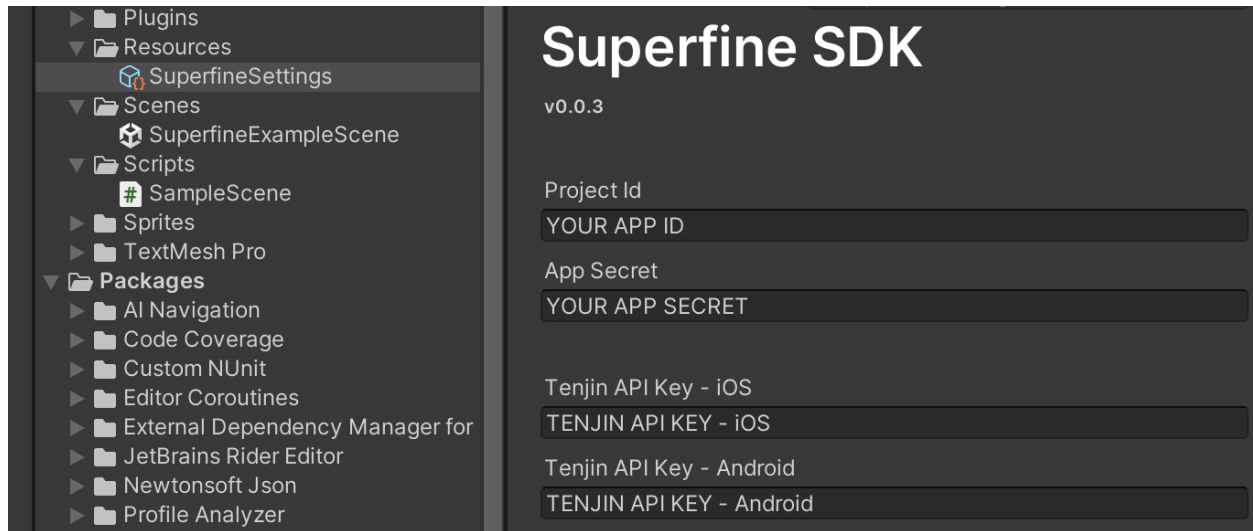
Save

## 1.3 Update Superfine Setting

From the menu pick **Superfine/Edit Settings**



Update your **Project ID**, **Superfine App Secret**, **TenjinApiKey** for **iOS** and **Android**



## 1.4 Initialize SDK

Add code to initialize the SDK (could be placed in the Awake function of a new component).

```
void Awake()
{
    TrackingManagerInitOptions options = new TrackingManagerInitOptions();

    #if !UNITY_EDITOR
    #if UNITY_ANDROID
        options.logLevel = LogLevel.VERBOSE;
    #elif UNITY_IOS
        options.debug = true;
        options.captureInAppPurchases = true;
    #endif
    #endif
    // Create instance
    TrackingManager.CreateInstance(options);
}
```

You can disable the automatic startup of the SDK from the configuration.

```
// Set autoStart to false
options.autoStart = false;
```

And then you can start it whenever you like.

```
TrackingManager.GetInstance().Start();
```

## 2 Send Events

### 2.1 Wallet Events

Call this event when you want to link the user wallet address:

```
TrackingManager.GetInstance().TrackWalletLink(wallet_address, "ronin");
```

Call this event when you want to unlink the user wallet address:

```
TrackingManager.GetInstance().TrackWalletUnlink(wallet_address, "ronin");
```

### 2.2 Game Level Events

Call this event when starting a level:

```
TrackingManager.GetInstance().TrackLevelStart(level_id, level_name);
```

Call this event when completing a level:

```
TrackingManager.GetInstance().TrackLevelEnd(level_id, level_name, true);
```

Call this event when failing a level:

```
TrackingManager.GetInstance().TrackLevelEnd(level_id, level_name, false);
```

### 2.3 Ads Events

These events are used to track ads from your app. You can use the Superfine dashboard later to check ad performance based on these events.

Call this event when an ad placement is loaded:

```
TrackingManager.GetInstance().TrackAdLoad(ad_unit, ad_placement_type,  
ad_placement);
```

Call this event when an ad is closed:

```
TrackingManager.GetInstance().TrackAdClose(ad_unit, ad_placement_type,  
ad_placement);
```

Call this event when the user clicks on an ad:

```
TrackingManager.GetInstance().TrackAdClick(ad_unit, ad_placement_type,
```

```
ad_placement);
```

Call this event when an ad is displayed:

```
TrackingManager.GetInstance().TrackAdImpression(ad_unit, ad_placement_type,  
ad_placement);
```

## 2.4 IAP Events

These events are used to track in-app purchases from your app.

Call this event when the user attempts to buy an IAP item:

```
TrackingManager.GetInstance().TrackIAPBuyStart(pack_id, price, amount,  
currency);
```

Call this event when the IAP purchase process is completed:

```
TrackingManager.GetInstance().TrackIAPBuyEnd(pack_id, price, amount,  
currency);
```

Call this event when restoring a purchase:

```
TrackingManager.GetInstance().TrackIAPRestorePurchase();
```

## 2.5 Custom Events

You can define any custom event to fit your needs.

Call this to send the custom event:

```
TrackingManager.GetInstance().Track(string event_name)
```

You can also create a class extending from TrackBaseData to store data for the event:

```
[Serializable]  
public class YourCustomEventData : TrackBaseData  
{  
    public string ip_address;  
    public string country;  
    public string device;  
    public string os_version;  
    public string app_version;  
    public string nft_id;
```

```
public int nft_ammount;  
public string chain_id;  
}
```

Call this to send the custom event with the custom data

```
TrackingManager.GetInstance().Track(string event_name, TrackBaseData data =  
null)
```