**Developer setup guide for Simple Payment Storefront Unity 3D asset (V2.0)**

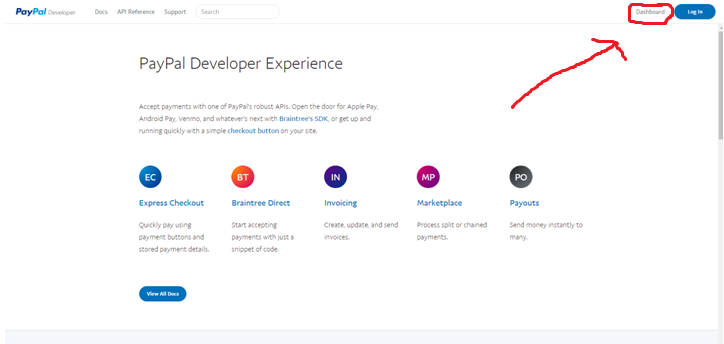
This document is split into two guides, setting up your PayPal account (Part 1) and configuring the asset in unity (Part 2).

**Part 1 - Setting Up Your PayPal Developer Account**

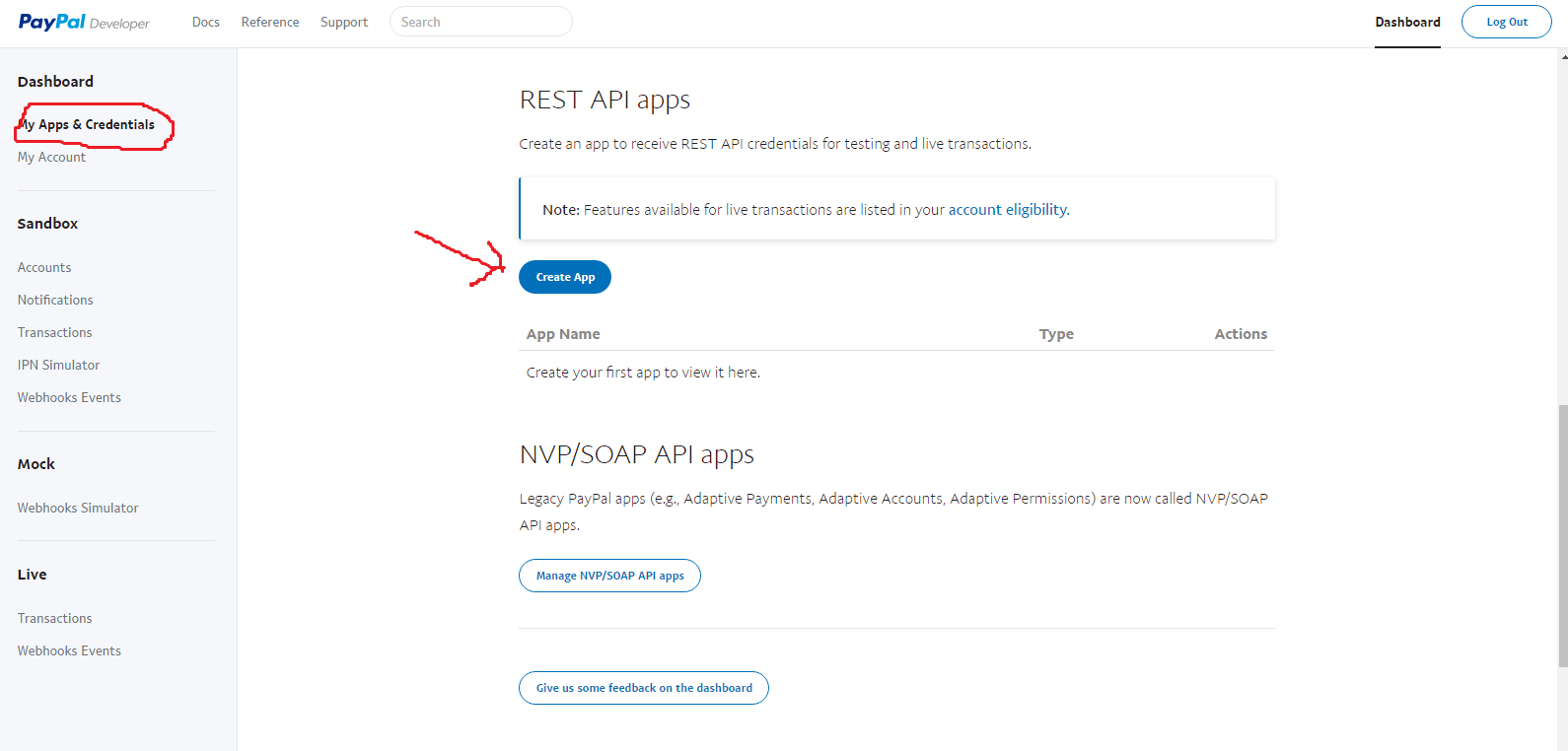
To accept payments from your players, you will need to create a PayPal developer account then login to the PayPal developer website and create a PayPal "App". This will give you a "clientID" and "secret" required within the unity asset for calling PayPal's REST APIs.

**Steps**

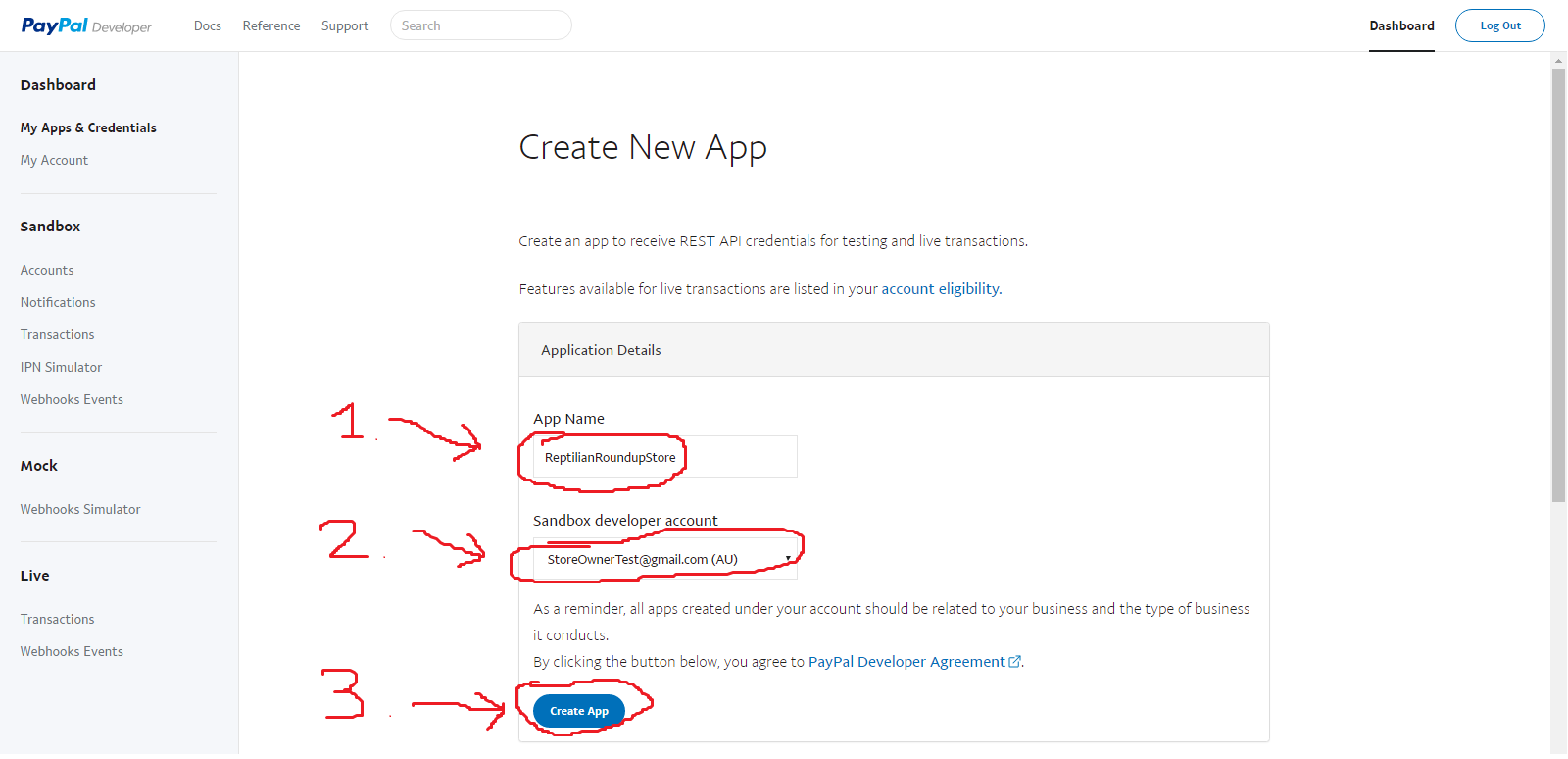
1. Login to the PayPal developer website: https://developer.paypal.com/
2. Click the dashboard link next to the logout button in the top right corner



1. Scroll down to the REST API's section of the "My Apps and Credentials" page and click the "Create App" button.

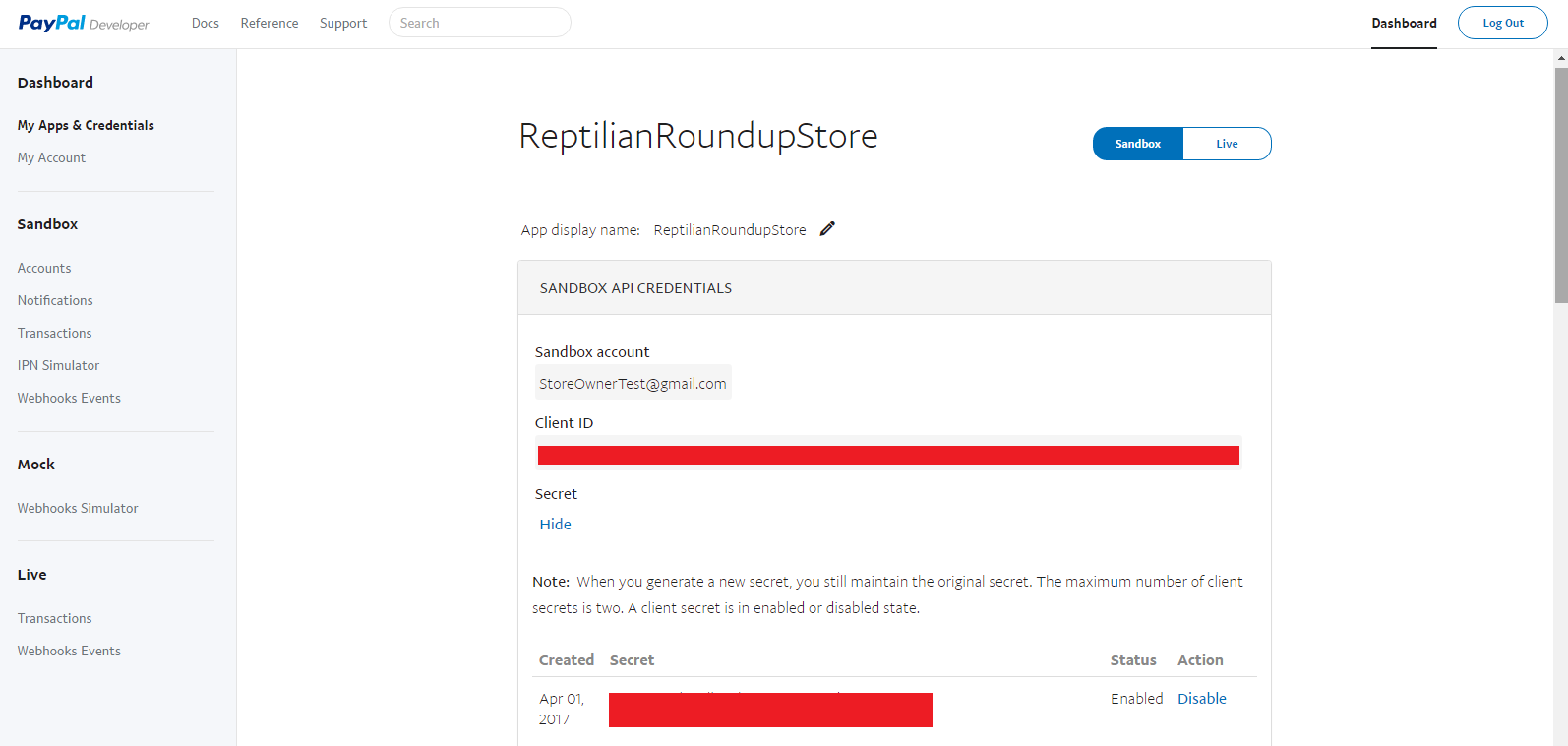


1. Enter a name for your app (such as the name of your game suffixed with "Store") select a Sandbox account then click the Create App button. (You can create additional sandbox accounts by clicking on the Sandbox -> Accounts link in the left menu.)



1. That's it! Now you should have a clientID and secret which you set as properties in this Unity asset as explained later.

Note you will have a different clientID and secret for live and sandbox endpoints. Click the Sandbox and Live buttons to toggle between displaying credentials for each account type.



**Part 2 - Configuring The Asset**

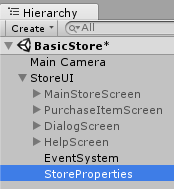
**Step 1 – Add the StoreUI to your game**

Add the “BasicStore” scene to your game’s build which is included within this asset and located under Assets/Store/Scenes/BasicStore

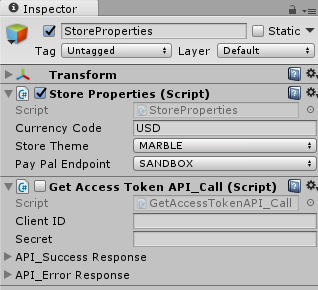
****

**Step 2 - Set Store Properties**

In the Game Object hierarchy, select the "StoreProperties" game object (located under StoreUI).

****

The store properties and PayPal API credentials can now be set by modifying the following values in the Inspector:



**Store Properties:**

These are the global properties for your store which are applied to each transaction:

**A) Currency Code**This is a three character currency code used to determine which type of currency your store will use. You should use a PayPal currency code associated with your PayPal account. For a list of valid currency codes see the following link:  
https://developer.paypal.com/docs/classic/api/currency\_codes/

**B) Store Theme**This value determines which background is used for the store.

**C) Pay Pal Endpoint**

This value can be set to either SANDBOX or LIVE and will determine whether the store re-directs players to the sandbox or the live PayPal checkout screen.

*The SANDBOX option allows purchases to be made without using real currency and should be used for developer testing.*

*The LIVE option should be used for live builds when you have implemented your game's item functionality and you're ready to start accepting payments from your players.*

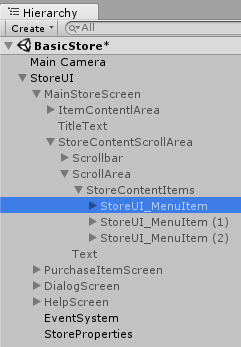
**Get Access Token API Call Properties:**

You need to set the "ClientID" and "Secret" properties to the values you're provided with upon creating your PayPal App. See Part 1, Step 5 of this guide for more details.

**Step 3 - Create store item content**

Now that the main properties for your store have been set up, it is time to start adding items to the store. The simplest way to do this is to duplicate an instance of the "*StoreUI\_MenuItem*" Game Object (select it in the hierarchy then press Ctrl+D).

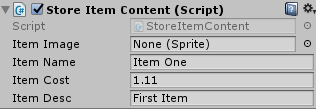
The below diagram illustrates where this object is located in the Game Object hierarchy.



**Ctrl+D**

Each instance of "*StoreUI\_MenuItem*" represents an item that will appear in your store.

The properties for each store item can be set by selecting a "*StoreUI\_MenuItem*" Game Object in the Hierarchy then editing it's StoreItemContent fields in the inspector. (See below diagram)

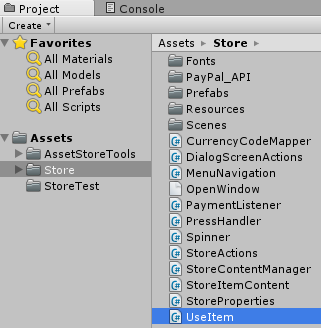
****

**Item Image (Optional)**  
This is the image you want to display in the store for the item. You may leave this field set to "None (Sprite)" to use the default image. All item image files need to be stored in the directory "Assets/Store/Resources/ItemSprites". After importing your images, you will need change their "Texture Type" property to "Sprite (2D and UI)" in the inspector.  
**Item Name (Required)**  
Item names should match the filename of the Item image WITHOUT the file extension. For example, if you have a sprite called "Golden Bananas.jpg" your item name should be "Golden Bananas". If you don't follow this naming convention your sprites won't appear correctly.  
**Item Cost (Required)**Enter the cost you want this item to be sold for (this should to be a value between 0.01 and 99.99)  
Float values will automatically get converted to the format (X.XX) when displayed in the store.  
**Item Description (Required)**Enter the description you want the player to see for the item in the store.

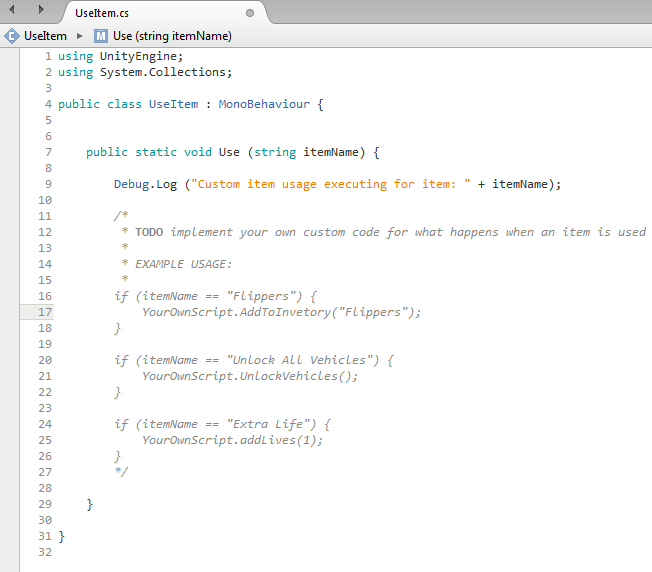
**Step 4 - Implementing the "Use()" function for items**

The basic configuration of your store is now complete. You can run your store within the Unity 3D editor (or create a build) and start making purchases.

At this point you should have a store setup where players can purchase items. In this step we will write code to implement what happens when the purchase of an item is completed. To do this, open the UseItem.cs script which can be found under Assets/Store/UseItem.cs



The *void Use(itemName)* is a callback method which is called whenever an item is successfully purchased. One parameter (itemName) is passed to the use method to identify which item has been purchased. It is up to you to write whatever you want to happen when specific items are purchased by players.



**Tip:**  A good way to implement the Use(itemName) method is to write all the actual item implementation code in your own separate scripts then just make a simple one line call to execute your own scripts from the Use(Itemname) method as seen in the above sample code.

**Optional Step 5 - Changing the "Purchase Complete" and "Purchase Cancel" redirect URLs**By default the player is redirected to https://www.google.com when a PayPal payment is completed or cancelled in the web browser. You can change these values in the Assets\Store\Resources\Misc\CreatePaymentRequestBody.json file.