**Developer setup guide for Simple Payment Storefront Unity 3D asset (V3.1.0)**

**Step 1: Setting up your PayPal account**

**Step 2: Testing your PayPal account integration in Unity**

**Step 3: Using the StoreFrontDemo scene**

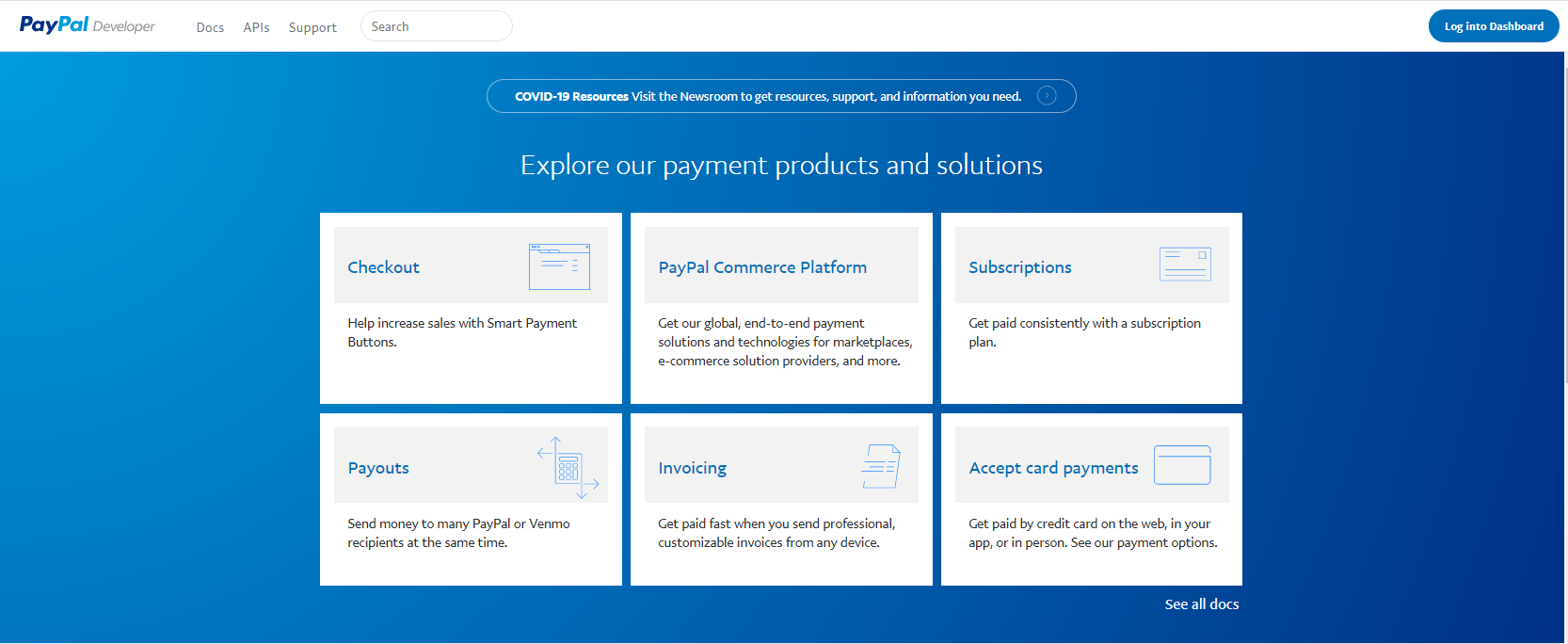
**Step 4: Technical overview of the PayPal API**

**Integration tutorial now available here:** <https://www.youtube.com/watch?v=0GxBi_zfyY8>

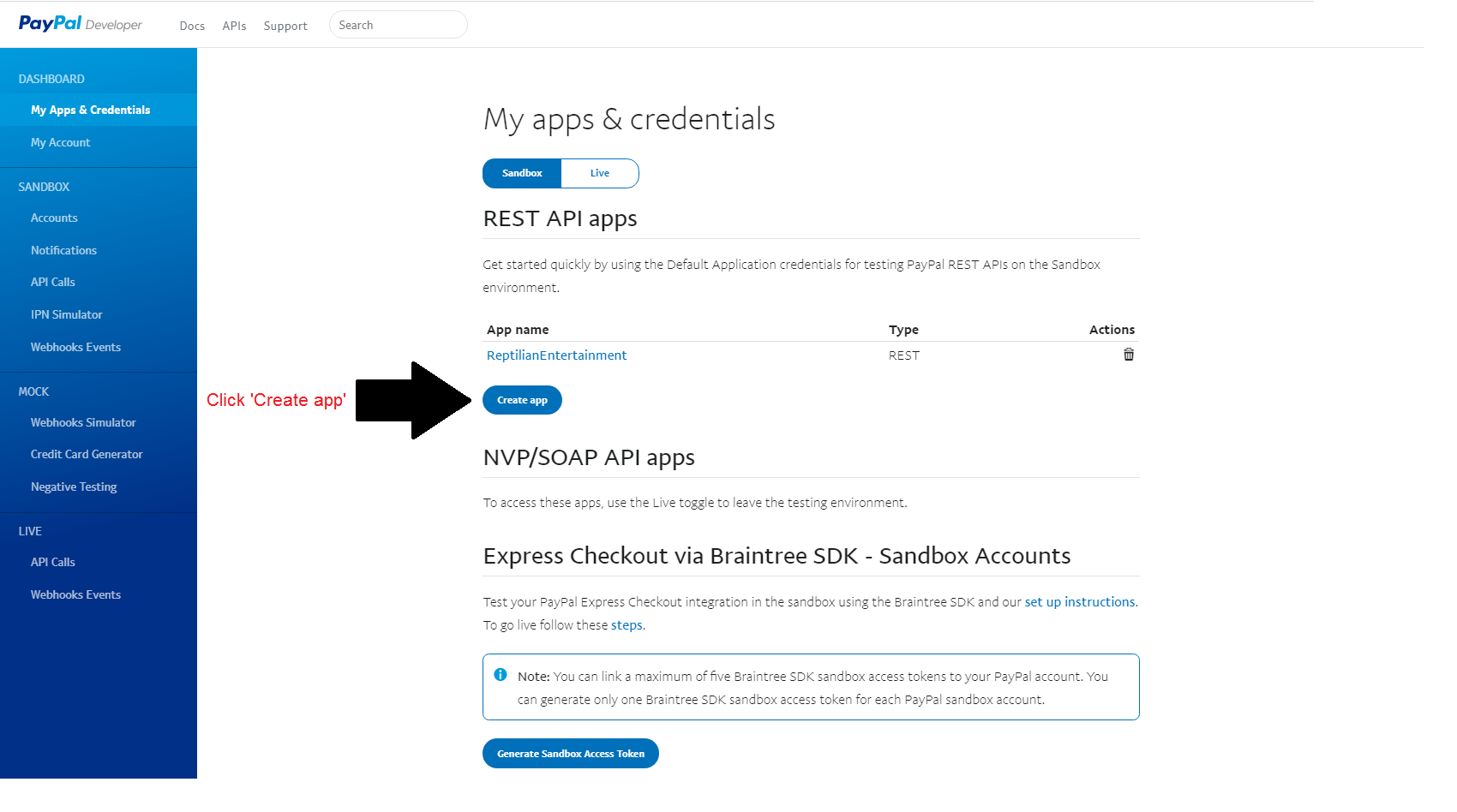
**Step 1 - Setting Up Your PayPal Developer Account**

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.

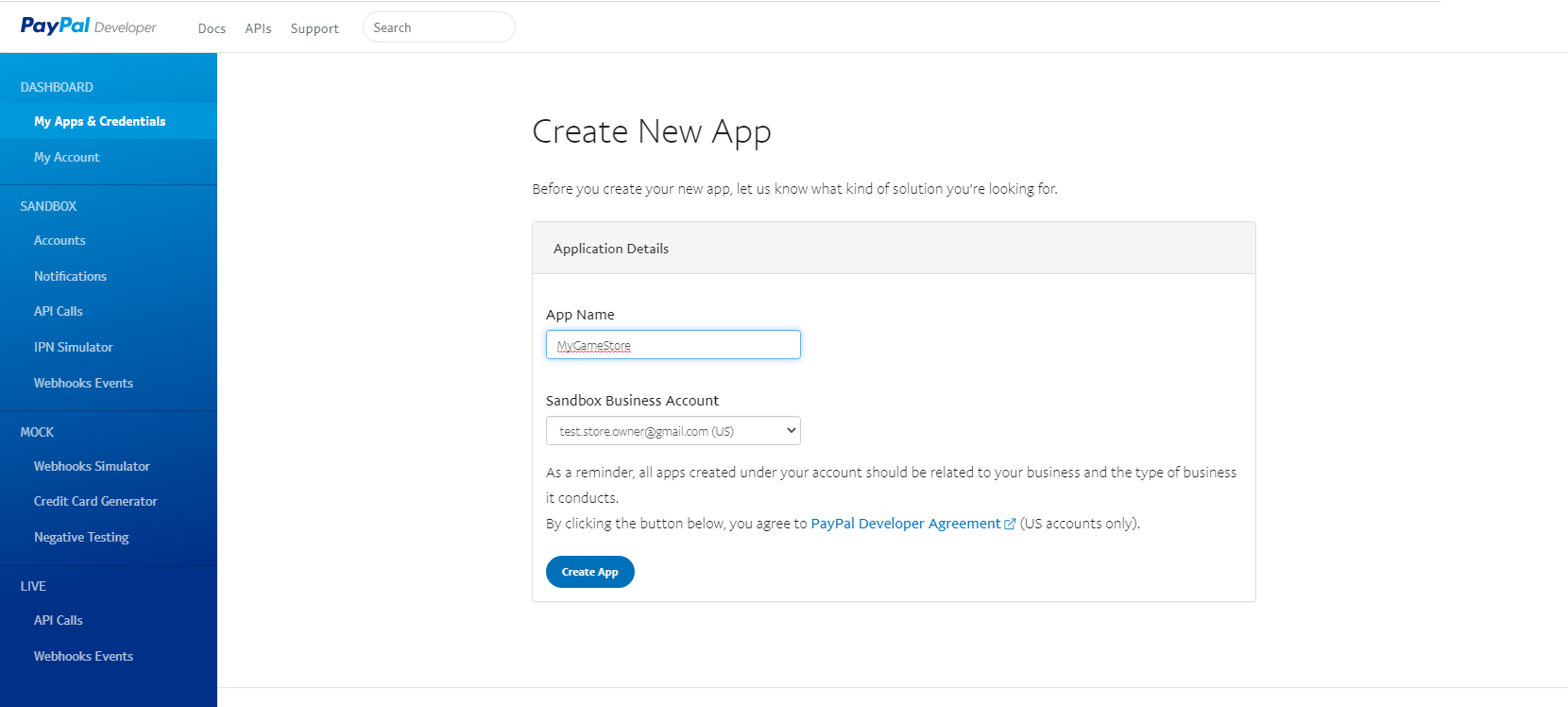
1. Go to the PayPal developer website: https://developer.paypal.com/classic-home/
2. Click the **Log into Dashboard** button in the top right corner then login



1. Click the **Create app** button under the REST API apps section

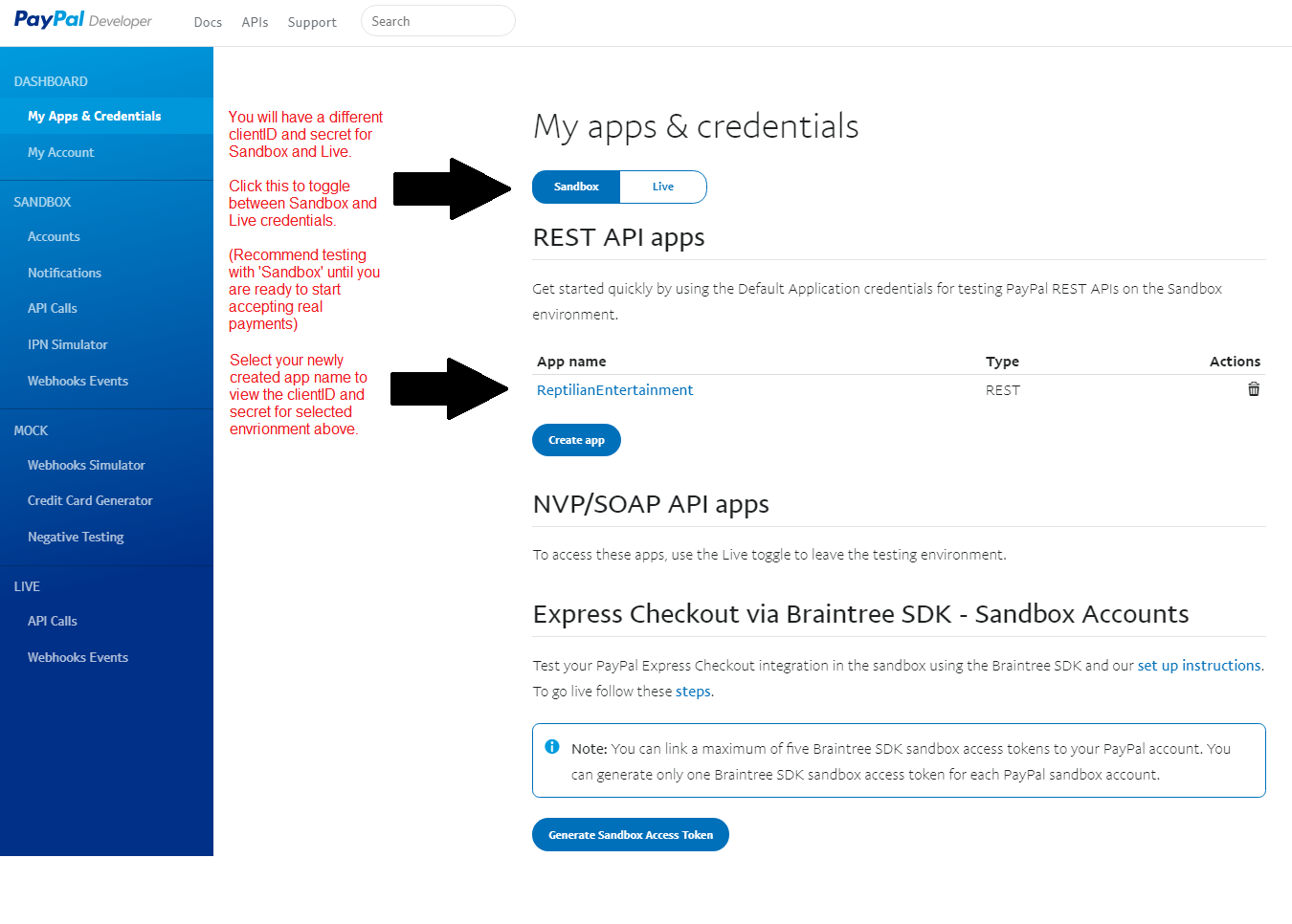


1. Enter a name for your app such as the name of your game suffixed with "Store", select a Sandbox account then click the Create App button. You can create additional sandbox accounts by clicking on the Sandbox -> Accounts link in the left menu.



1. That's it! You'll now have a clientID and secret for calling both sandbox and live PayPal endpoints.

You can view these by going back to the PayPal developer dashboard then selecting the hyperlink for your newly created application.



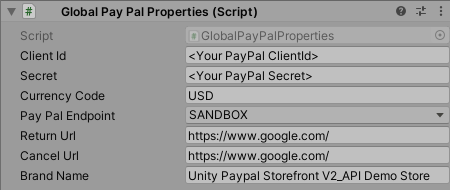
**Step 3 - Testing your PayPal account integration in Unity**

The BarebonesUI\_Demo\_V1 and BarebonesUI\_Demo\_V2 scenes demonstrate the basic flow of a PayPal transaction in Unity. You can use these scenes to test your PayPal developer credentials and learn the basics on how to make calls to the PayPal API.

The main difference between the V1 and V2 barebones scenes is that V1 uses the deprecated PayPal APIs and V2 uses the latest APIs. The V2 scene also has a more advanced UI which contains a basic an interactive tutorial whereas the V1 scene is very minimalistic. Testing with the V2 scene and developing with the V2 PayPal API is recommended.

**Running the Barebones Demo Scenes**

In order to run either of the V1 or V2 demo scenes, you'll need to set the **ClientId** and **Secret** on the **GlobalPayPalProperties** script object (You may also want to change the other default values). A single instance of this script exists in each of the barebones scenes under the \_SCRIPT\_INSTANCES game object in the hierarchy:



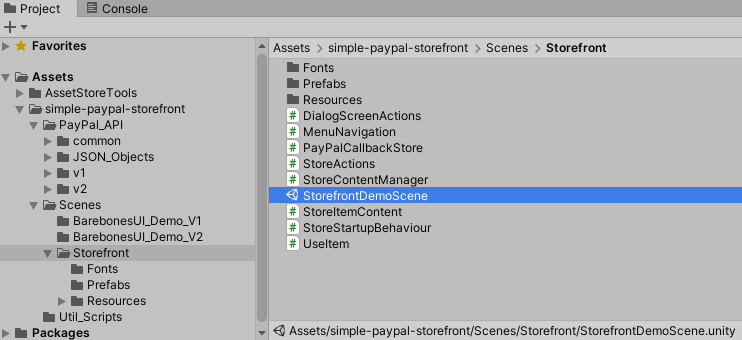
Now you can run the scene and you should be able simulate a transaction. From here you can build your own UI and transaction management solution based off the PayPal API calls made within this scene. (Or you can go to Part 3 of this document which explains how to use the StorefrontDemoScene which provides a "ready-to-go" simple storefront).

**Part 3 - Using the StoreFrontDemo scene**

The StoreFrontDemo scene is a lightweight end-to-end demo of the PayPal API that provides basic transaction management and item usage functionality within a simplistic UI. Feel free to import this storefront directly into your game however it's recommended that you should build your own storefront and only use the PayPal API scripts included in this asset for more serious projects.

**Note**: This scene was developed using the deprecated V1 PayPal API. The V1 API still works fine however a new storefront scene using the V2 API is currently in development for this asset which is targeted for a late 2020 release.

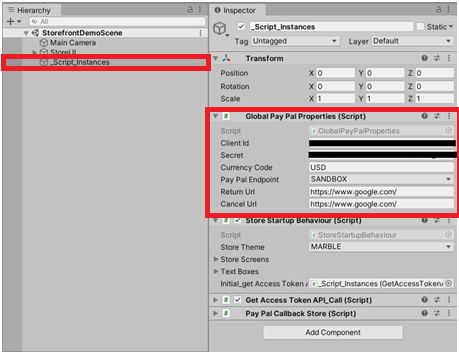
**Step 1 – Add the StoreFrontDemo scene to your game**

****

Add the **StorefrontDemoScene** scene to your game’s build which is included within this asset and located under:   
simple-paypal-storefront/Scenes/Storefront

**Step 2 - Set Store Properties**

In the Game Object hierarchy, select the **\_Script\_Instances** game object.



**Global Pay Pal Store Properties:**

These are the global store properties and PayPal API credentials used for each transaction. Each property needs to be set as follows:

**A) ClientId and Secret**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.

**B) 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/

**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.*

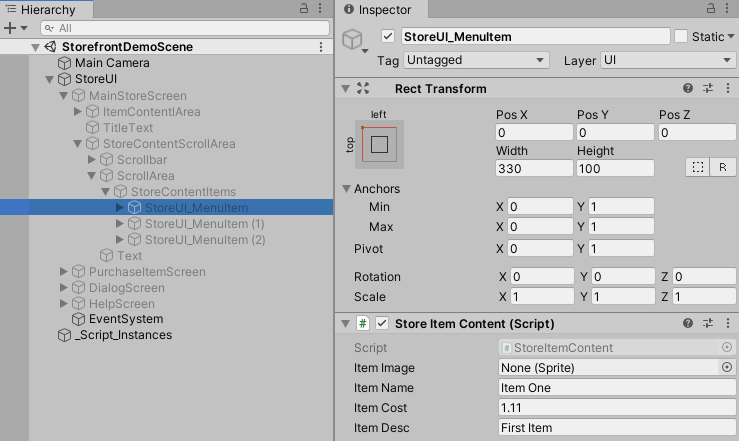
*The LIVE option should be used for live builds when you have implemented your game's item functionality and you're ready to start accepting payments from your players.*

**D) Return URL and Cancel URL**These are the URLs the user will be re-directed to when the user completes or cancels a transaction.

**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).

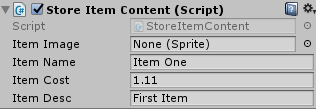
The below diagram illustrates where this object is located in the Game Object hierarchy.



**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 "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 "3 month subscription.jpg" your item name should be "3 month subscription". 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.

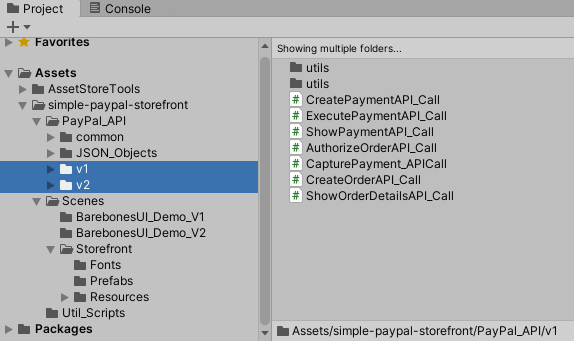
At this point you should have a store setup where players can purchase items. In this step we will write code to implement what happens when the purchase of an item is completed. To do this, open the UseItem.cs script which can be found under StorefrontDemo/UseItem.cs

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.

**Part 4 - Technical overview of the PayPal API**

The core PayPal API functionality of this asset lives within PayPal\_API v1 and v2 directories.

****

This asset makes use of two sets ofPayPal endpoints to facilitate a transaction with the player: the V1 and V2 PayPal endpoints. Originally this asset was developed using only the V1 endpoints, however PayPal has since released V2 endpoints so this asset has been updated to implement both versions.

Further documentation on these PayPal endpoints can be found here:   
V1 documentation: <https://developer.paypal.com/docs/api/payments/v1/>  
V2 documentation: [https://developer.paypal.com/docs/api/payments/v2/](https://developer.paypal.com/docs/api/payments/v1/)

**Overview of PayPal endpoint API\_Call classes**

Each of the ...API\_Call scripts in this asset have their own public properties that need to be set before they are instantiated.

Since each of API\_Call scripts implement MonoBehaviour, they need to be instantiated in the same way MonoBehaviour objects are typically instantiated.

Examples of this can be seen in both of the demo scene scripts where the API\_Call scripts are instantiated via calling addComponent() on a game object.

**Basic flow of the PayPal API process V1:**

1. Obtain the *access\_token* by calling **GetAccessTokenAPI\_Call** (this token will be passed into each subsequent API call).
2. Create a payment by calling **CreatePaymentAPI\_Call**  and passing it the previously obtained access token. This will return a unique payID.
3. Keep calling **ShowPaymentAPI\_Call** periodically with the previous payID until it returns a "VERIFIED" status with a payerID.
4. Finally call **ExecutePaymentAPI\_Call** with the payID and payerID to complete the transaction.

**Basic flow of the PayPal API process V2:**

Run the BarebonesUI\_Demo\_V2 scene for a tutorial on the V2 API

**Note:**

The **GetAccessTokenAPI\_Call** is common to both V1 and V2 APIs, it can be found at the following location: 

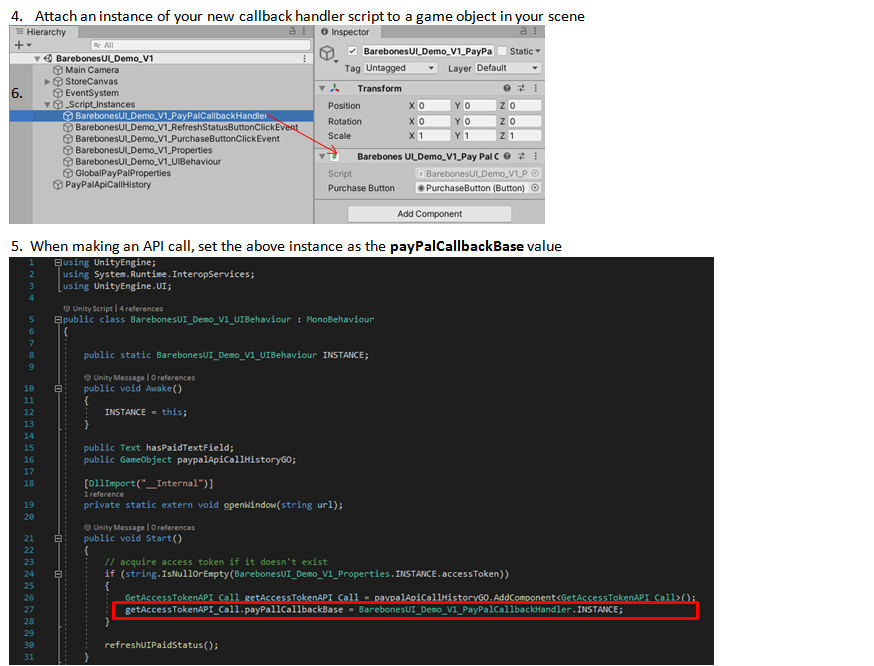
**PayPal API callback handling overview**

The **PayPalCallbackBase, PayPalCallbackBase\_V1 and PayPalCallbackBase\_V2** classes provide an easy way to handle the responses received from making PayPal API calls. These abstract classes contain abstract methods that represent the PayPal endpoints.

Each scene in this asset has it's own implementation of the callback handler classes.

The below diagrams illustrate how these callback classes operate in the BarebonesUI\_Demo\_V1 scene (Note that the **PayPalCallbackBase\_V2** class is not used in the BarebonesUI\_Demo\_V1 scene but it's still included in the below diagram for completeness):





\* The authorizeOrderSuccess() API call method in the PayPalCallbackbase\_V2 class is only needed as an extra step for orders that are created with the AUTHROIZE\_PAYMENT intent value set. By default the intent property is left blank on the create order API call which means the authorize order API call can be skipped.

See the PayPal documentation for more details: https://developer.paypal.com/docs/api/orders/v2/#orders\_create