**Developer setup guide for Simple Payment Storefront Unity 3D asset (V1.1)**

This guide provides a walkthrough on how to set up the Simple PayPal Storefront asset in your game.

In order to accept payments from your players, you will need to have a PayPal account setup that is capable of receiving payments. Please make sure you have one of these accounts setup before continuing with this guide.

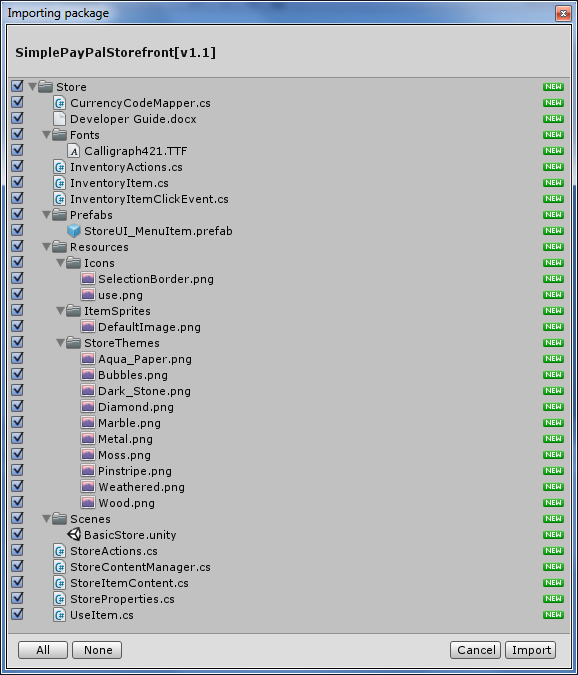
You can create a new PayPal account here: https://www.paypal.com/us/signup/account

For details about the revenue model of this asset, see this link: https://drive.google.com/file/d/0BzxKCQUuh2SvRDJ0dTJ3WWRVMmM/view?usp=sharing

**Step 1 - Download and Import the asset into your project**

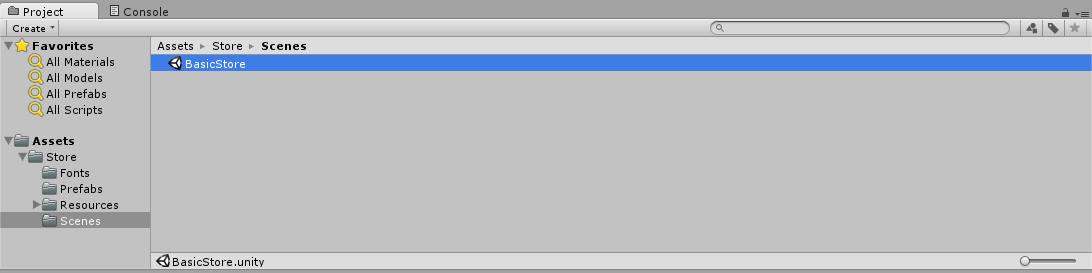
Download the In-game PayPal Store asset from the Unity 3D asset store.

Open your project, or a create a new Unity 3D project and import the In-game PayPal Store asset into your project (make sure all asset components are selected).



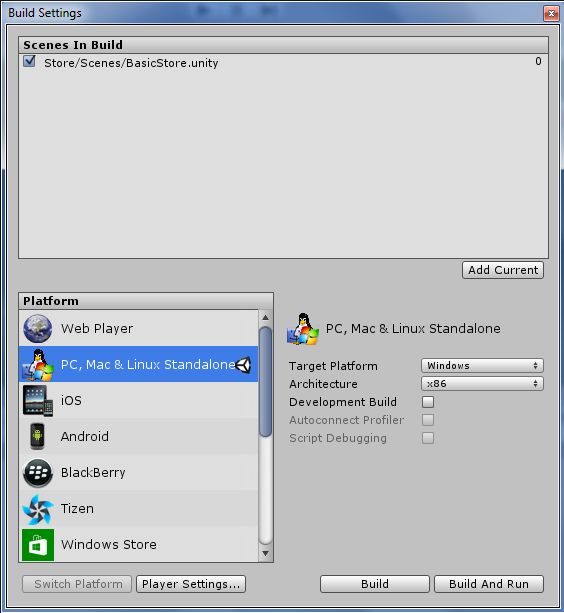
**Step 2 - Add BasicStore scene to build scenes**

Open the "BasicStore" scene that has been imported into your project. This scene is located in the Assets\Store\Scenes directory.



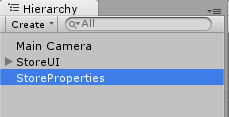
In the unity editor press Ctrl+Shift+B then click "Add Current" to add the BasicStore scene to the project's build scenes.

(You don't need to build your project now so you can click the X after clicking the Add Current button).

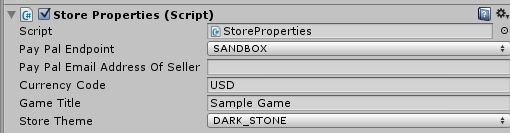


**Step 3 - Set Store Properties**

In the scene hierarchy for the "BasicStore" scene, select the "StoreProperties" game object.



The properties for your store can now be set by modifying the values in the Inspector for the StoreProperties script.



As seen in the above figure, there are *five* properties that need to be set. Instructions are given below for how each property should be set.

**Pay Pal Endpoint**

This value can be set to either SANDBOX or PRODUCTION 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 production 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.*

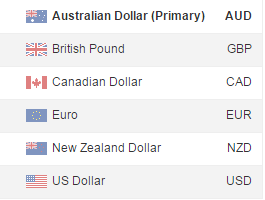
**Pay Pal Email Address**

This is the email address associated to the PayPal account that will receive payments from purchases made in your store. As mentioned as the beginning of this document, If you don't already have one these accounts set up, you will need to go to PayPal's website and set up a personal premier or business account so you can accept payments from your players.

*NOTE:* For sandbox testing you may leave this field blank to use the default sandbox seller email that has already been setup.

**Currency Code**

This is a three character currency code used to determine which type of currency your store will use. The following six currency codes are currently supported:



**Note:** Only Australian (AUD) and US (USD) currency codes will work when using the sandbox environment.

**Game Title**

This is the title of your game. It is also the name the players will see during PayPal's checkout process.

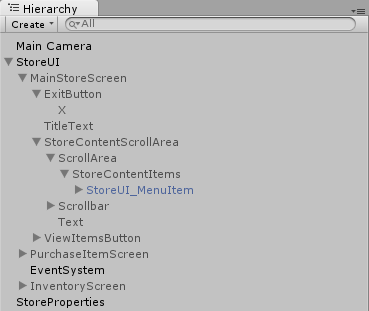
**Store Theme**

This value determines the which background is used for the store.

**Step 4 - 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 the "*StoreUI\_MenuItem*" Game Object (select it in the hierarchy then press Ctrl+D).

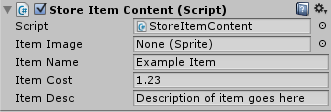
The below diagram illustrates where this object is located in the Game Object hierarchy.



**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)

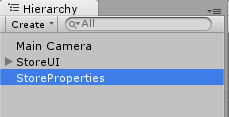
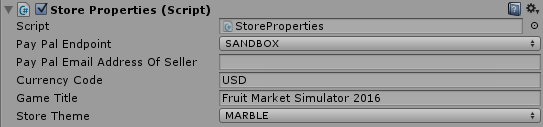
****

**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 (Mandatory)**  
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 (Mandatory)**Enter the cost you want this item to be sold for (this needs to be a float value between 0.01 and 99.99).  
Float values will automatically get converted to the format (X.XX) when displayed.  
**Item Description (Mandatory)**Enter the description you want the player to see for the item in the store.

**Step 5 (Optional Step) - Test purchasing of items in sandbox environment**

The basic configuration of your store is now complete. You can run your store within the Unity 3D editor (or create a build) to view what it currently looks like and test the purchasing of items using the sandbox environment.

*Note:* Make sure the PayPal endpoint field is set to SANDBOX on the StoreProperties Game Object

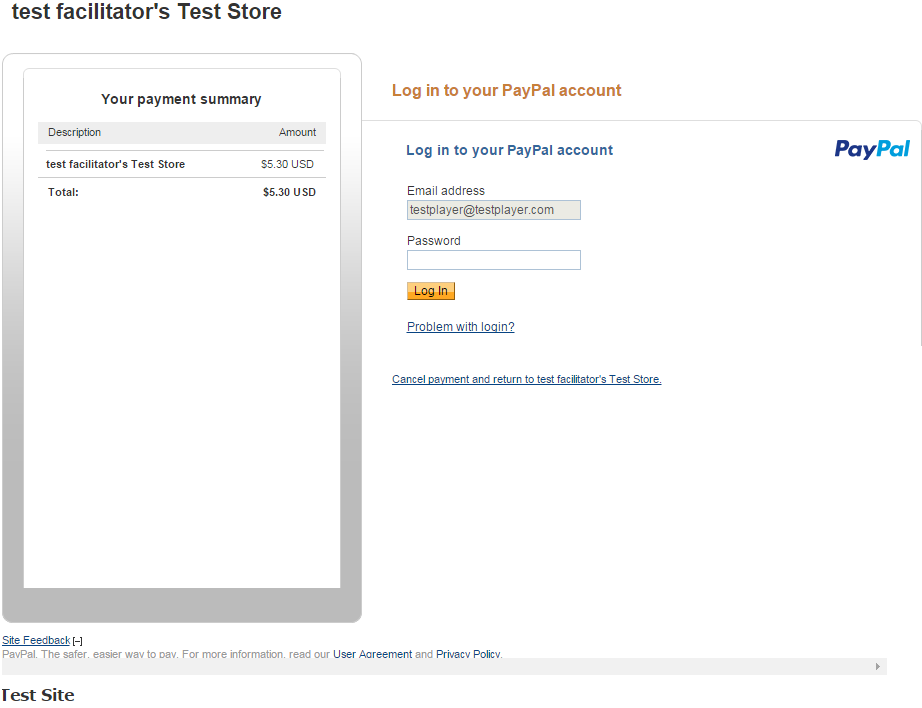
Run your project then click the "Buy" button for one of the items:



The player will now be taken to the "Purchase Item" screen which will be in a "Waiting" status

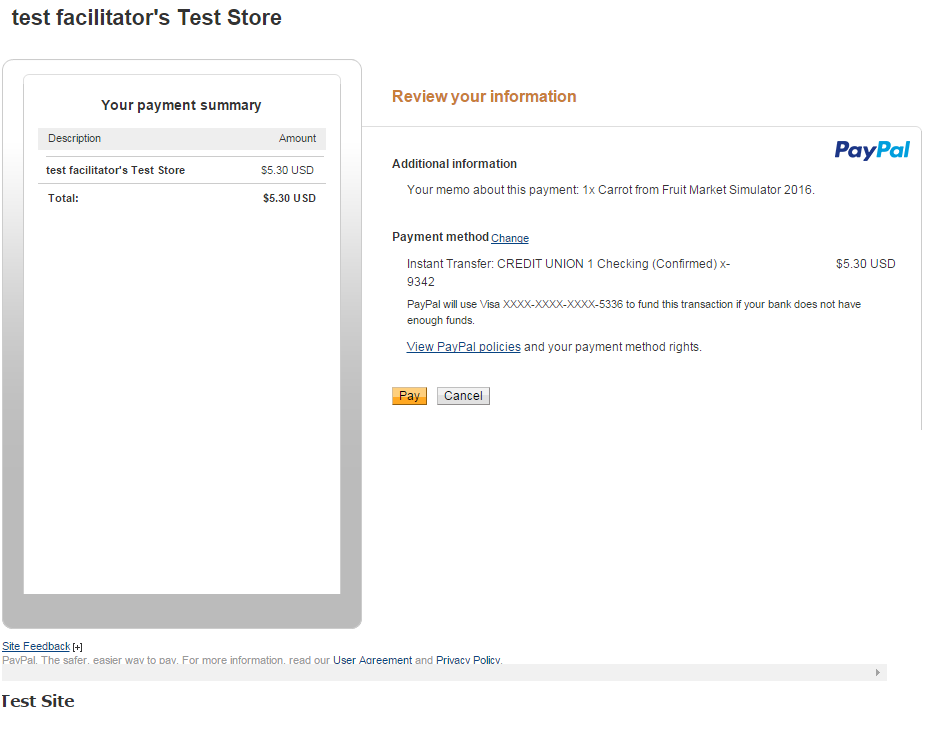


Meanwhile, a new browser window will be opened to complete the test payment

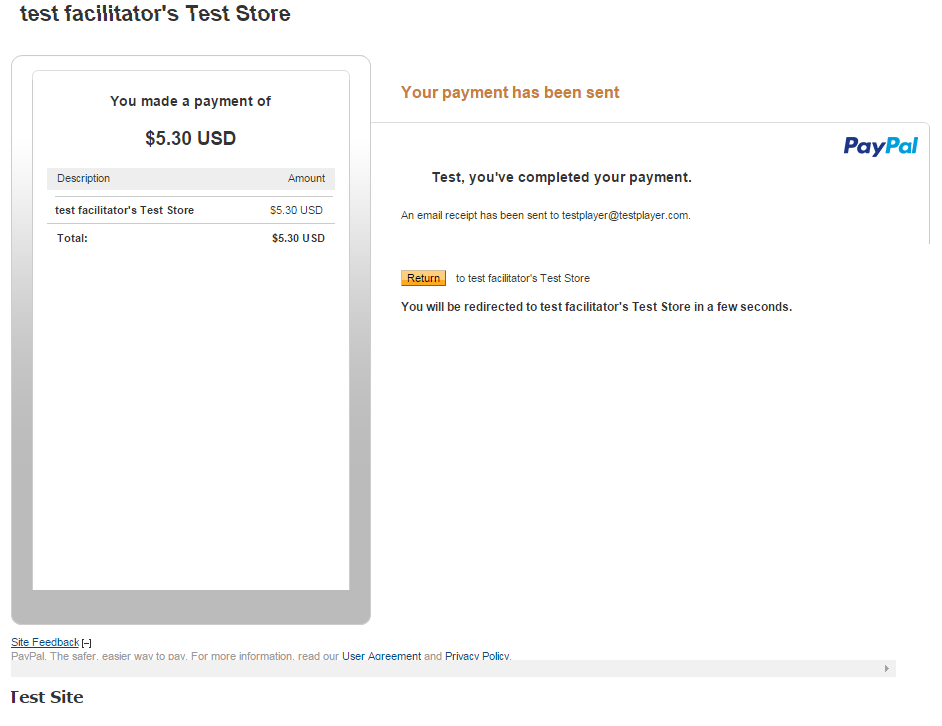


The default test buyer account (testplayer@testplayer.com) will be used to make the test payment. The password for this account is "**testplayer**".

**Note**: When using the PRODUCTION endpoint store property, the email address field will be blank and the buyer will need to supply their own PayPal credentials to complete the payment.

After entering the password and clicking the "LogIn" button, the following payment confirmation screen will be displayed. 

Click the "Pay" button to complete the test payment.



You can now close PayPal web browser window and return to your game. After a few seconds the in-game purchase status should change to "SUCCESS!"



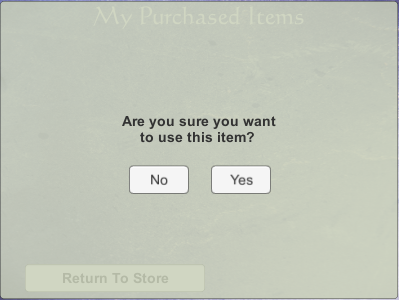
**Note**: You do not need to wait for the "SUCCESS" message to be displayed in order to receive your item . You will be able to view your new items immediately after they are purchased by clicking the "Return To Store" button then the "View Purchased Items" button.

Now you can open the "My Purchased Items" screen and you should be able to see the item you have just purchased.



This item can now be either used or deleted by clicking the respective green or red circular button icons.

When one of these buttons are clicked, a dialog will appear prompting the user whether they are sure they want to commit to their choice of using or deleting the selected item:



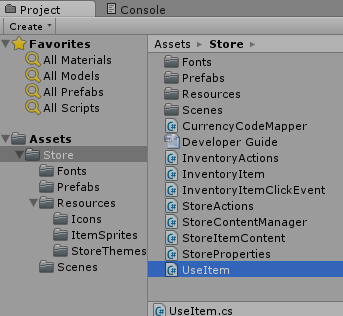
When an item is used, the inventory icon for that item will be grayed out and the status of the item will change from "Purchased" to "Purchased and Used". Items in this status can only be deleted and not be re-used (the use button will be disabled for used items).

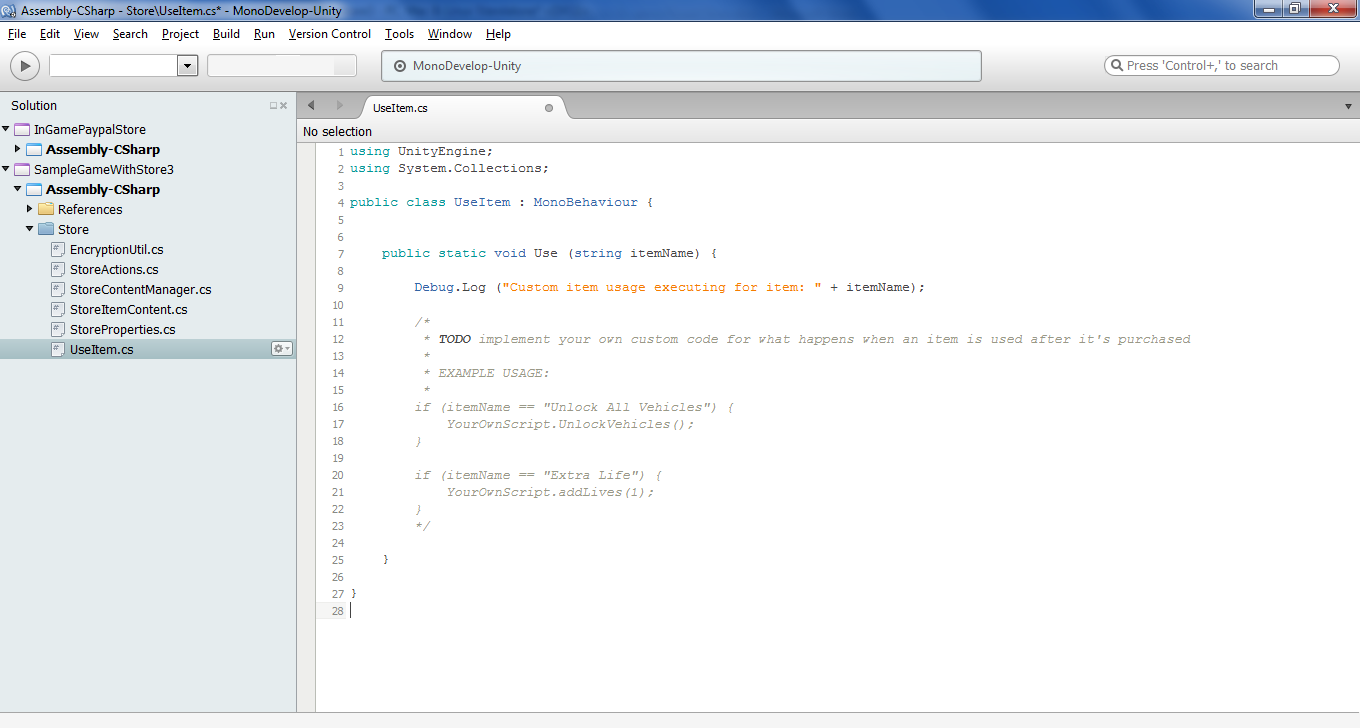
The next step of this guide will explain how to write custom code for when the "Use" button is clicked.

**Note:** Currently a maximum of 25 items can be displayed in the inventory. Each item has a priority based on its status and purchase date. The most recent un-used items will appear at the start of the inventory and oldest used items will appear at the end of the inventory. If a user has more than 25 items only the first 25 items with the highest priority will be displayed.

**Step 6 - Implementing the "Use()" function for items**

At this point you should have a store setup where players can buy, see, use and delete their items, however nothing happens when the player uses that item other than the item changing it's state from "Purchased to "Purchased and Used". In this step we will write code to implement what happens when the player uses specific items. To do this, open the UseItem.cs script which can be found under Assets/Store/UseItem.cs





The Use(itemName) is a callback method which is called whenever an item is used by a player. One parameter (itemName) is passed to the use method to identify which item is being used. It is up to you to write whatever you want to happen when specific items are used by players. See the above screenshot for some basic implementation suggestions.

**NOTE:**  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 sample code.

**Step 7 - Link the store scene to the rest of your game**

The final step is to implement where the store is initially loaded from and where the player is directed to when they close the store (i.e. when the player clicks the 'X' button in the top right corner of the store).

*Opening the Store*

Opening the store simply involves loading the "BasicStore" scene from another scene. For example, your game might have a MainMenu scene with an "Open Store" button and for the code of that button you would just need to write the following line to open the store:

Application.LoadLevel("BasicStore");

*Closing the Store*

The store is closed when the player clicks the X button on the top right corner of the main store screen. The code for this action is located in the CloseStore() method of the StoreActions.cs class (found in Assets/Store/StoreActions.cs). In this method you will most likely want to add a line of code that loads another scene, such as the main menu scene of your game.

For example:

