**Developer setup guide for Simple Payment Storefront Unity 3D asset (V1.2)**

This document will guide you on how to set up the Simple PayPal Storefront in your game.

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 PayPal transaction process of this asset, see this link: https://drive.google.com/file/d/0BzxKCQUuh2SvRDJ0dTJ3WWRVMmM/view?usp=sharing

For estimated profit margins on a range of sale prices, see this link:  
https://drive.google.com/file/d/0BzxKCQUuh2SvRHFXNV8ycjJRM00/view?usp=sharing

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

To use this asset, you will need to add an instance of the “StoreUI” prefab into a scene within your game. The “StoreUI” prefab object is located under Assets/Store/Prefabs/StoreUI

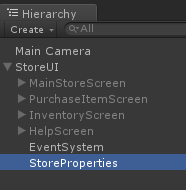


Alternatively, you can add the “BasicStore” scene to your game’s build which is included within this asset and located under Assets/Store/Scenes/BasicStore

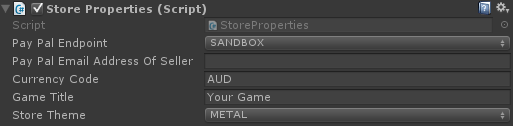
****

**Step 2 - Set Store Properties**

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

****

The 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.*

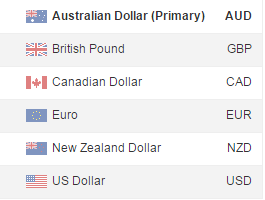
**Pay Pal Email Address Of Seller**

This is the email address associated to the PayPal account that will receive payments from purchases made in your store. As mentioned at 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 an account that can accept payments from your players.

NOTE: For sandbox testing you should leave this field blank to use the default sandbox seller email that has already been set up.

**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.

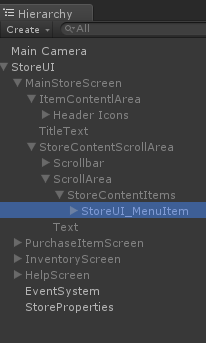
**Store Theme**

This value determines which background is used for the store.

**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).

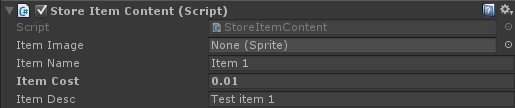
The below diagram illustrates where this object is located in the Game Object hierarchy.



**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)

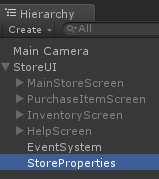
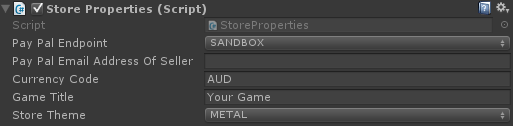
****

**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 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 in the store.  
**Item Description (Required)**Enter the description you want the player to see for the item in the store.

**Step 4 - 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

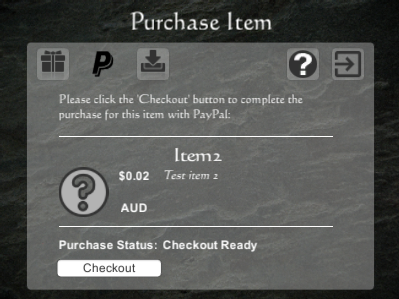
Run your project.

The store will load with the Main Store Screen displayed. This is the primary store screen that will display the items available for purchase. If you’re not already on this screen, you can navigate to this screen by clicking the present icon in navigation menu.



Click the “Buy” button for one of the items.

The player will now be taken to the "Purchase Item" screen where the player can checkout their previously selected item and view the purchase status of the selected item.



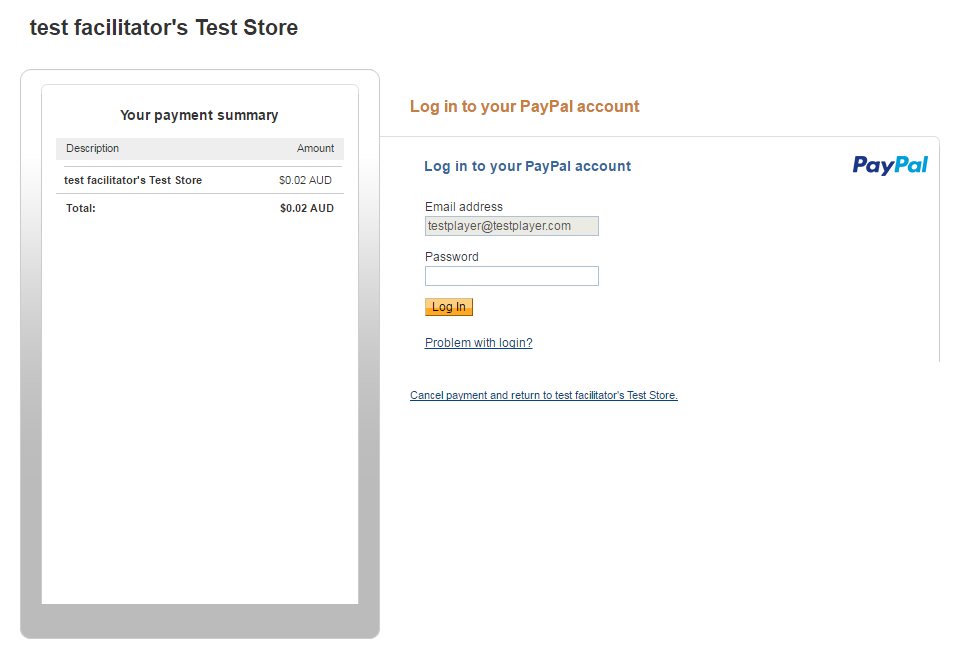
There are five purchase status states this screen can be in:

1. No Item Selected
2. Creating Purchase
3. Checkout Ready
4. Waiting
5. Purchase Complete

Shortly after clicking the “Buy” button for an item, the Purchase Item screen will enter the Checkout Ready state and the Checkout button will be displayed.

Click the Checkout button to open a new PayPal tab in your web browser.

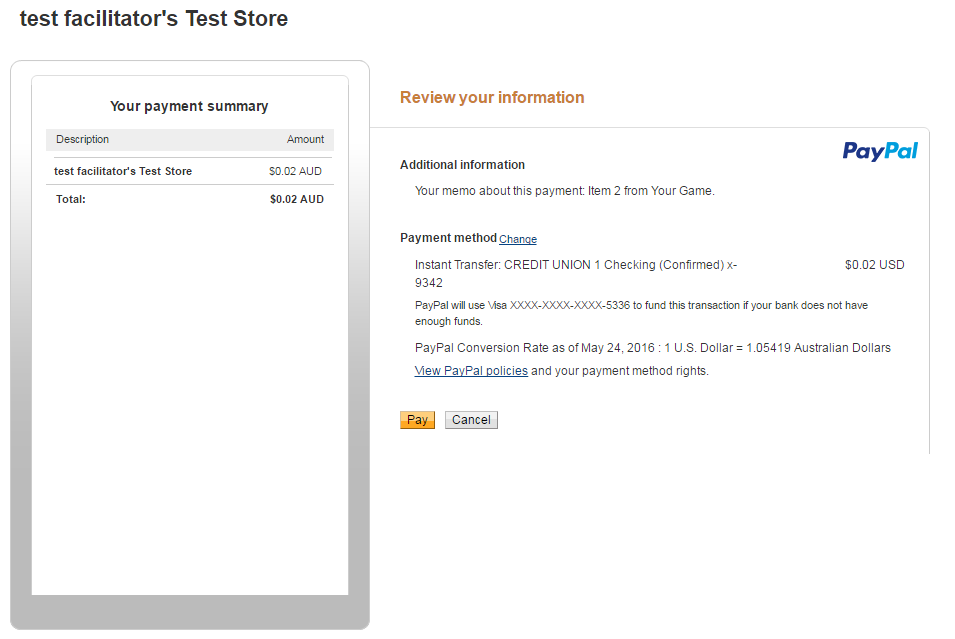
The following checkout page should now be open in your web browser:



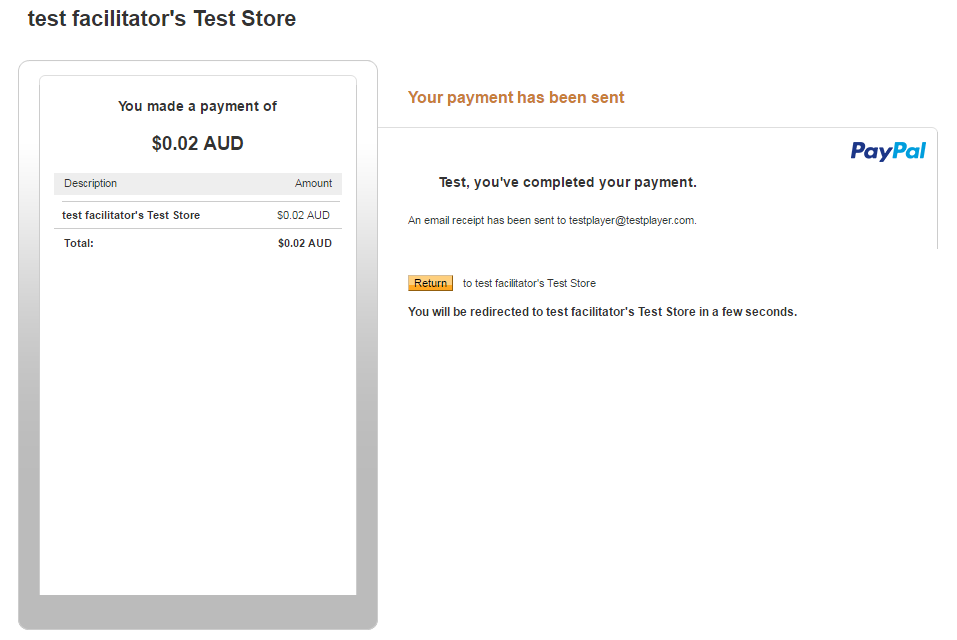
The default test buyer account (testplayer@testplayer.com) will be used to make the test payment.

The password for this account is "**testplayer**".

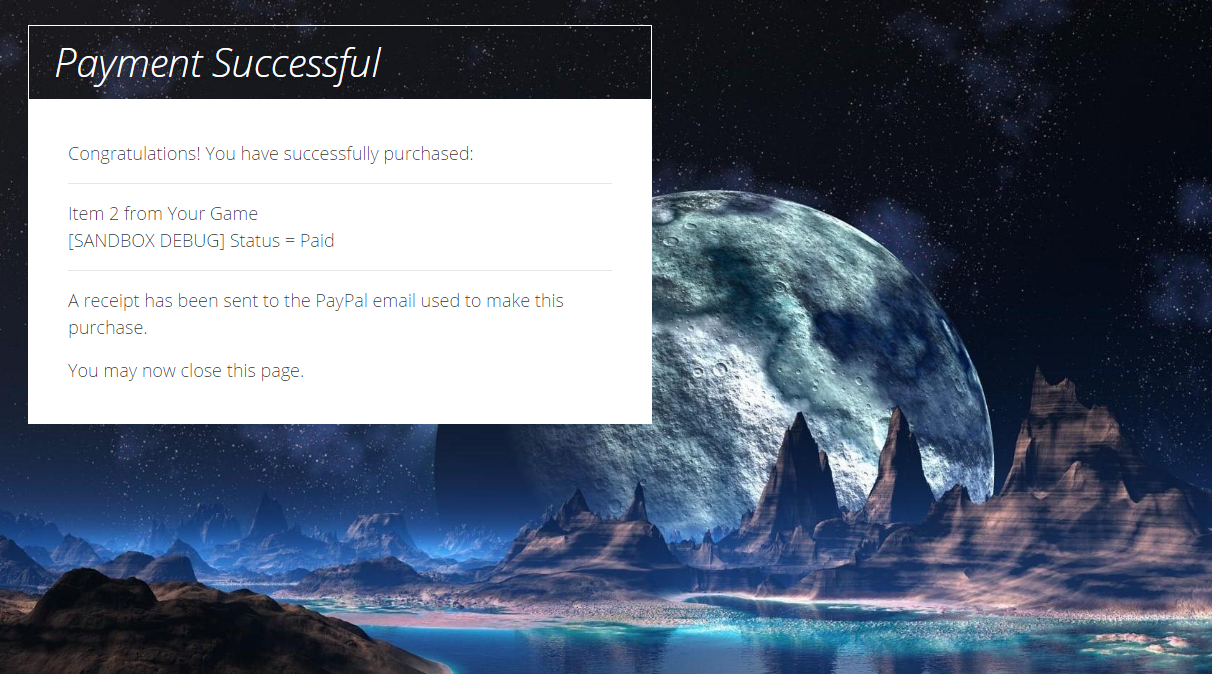
**Note**: When using the PRODUCTION endpoint store property, the email address field will be blank and the buyer will need to supply their own PayPal credentials to complete the payment.

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.

After clicking Pay, the payment confirmation screen will be displayed:

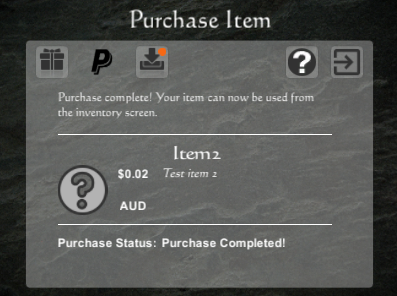
After a few seconds, you will be redirected to the Reptilian Entertainment Payment success page.



For live payments the user can safely close their browser (or tab) before being re-directed to the Reptilian Entertainment page. However for sandbox payments, you should wait for the Reptilian Entertainment Payment Successful page to load because there is a trigger on the page that will fast-track the server-side processing of the sandbox payment. Not letting this happen could result in your sandbox payment not being processed for a long time and result in your game's StoreUI getting stuck in the "Waiting..." purchase state.

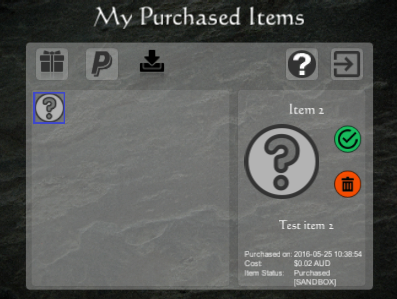
**Step 5 - View, Use and Delete Purchased items**

Once a payment has been completed, the purchase status state of your store will transition from "Waiting" to "Purchase Completed!":



An orange notification will also appear next to the store inventory icon to prompt the player to view their new item in their store inventory.

Clicking the store inventory icon will open the following screen:

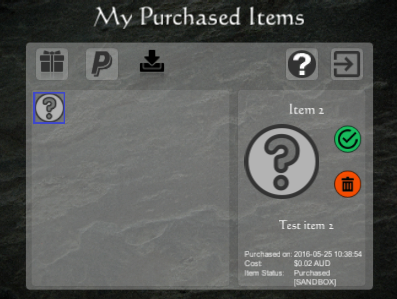


On this screen, items purchased will appear on the left and the item details along with their available item actions (use and delete) will be displayed on the right.

Use the mouse to point and click on the item images on the left panel to select different items. Doing this mark an item as selected which will be indicated as having blue square border.

Note: The most recently purchased item will be selected automatically when the inventory screen is loaded.

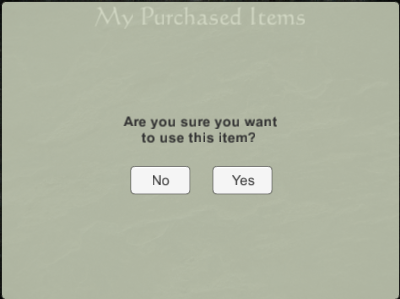
Items can be either used or deleted by clicking the respective green or red circular button icons.



**Delete Item**

**Use Item**

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:



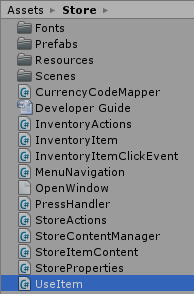
When an item is used, the inventory icon for that item will be grayed out and the status of the item will change from "Purchased" to "Purchased and Used". Items in this status can only be deleted and not be re-used (the use button will be disabled for used items).

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

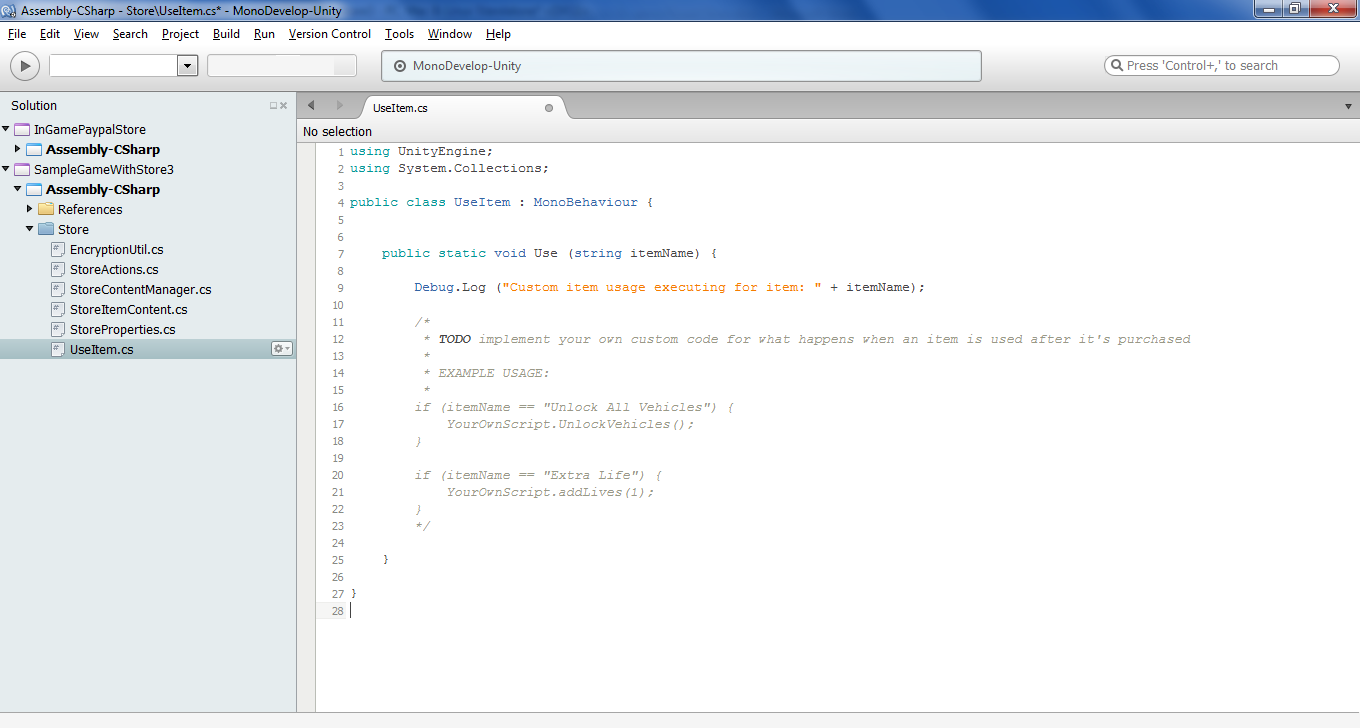
**Note:** Currently a maximum of 25 items can be displayed in the inventory. Each item has a priority based on its status and purchase date. The most recent un-used items will appear at the start of the inventory and oldest used items will appear at the end of the inventory. If a user has more than 25 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 *void 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. The comments in this class show some basic implementation suggestions:



**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 sample code.