WUMONEY QUICKSTART GUIDE

WUMoney is a virtual currency system made for use with the WUSS system. It features integration with Tapjoy to allow you to monetise your games by working in conjunction with the Tapjoy SDK for Unity, rather than as a replacement for it. Almost everything related to Tapjoy and it's SDK remains the same and you use it exactly the way you normally would have.

To that end most of what you are going to do is done via their plugin, explained on their website and used according to how they say. I have taken all the steps I can to ensure using WUMoney and using Tapjoy directly feel the same as much as possible so that using the two kits together feel as natural as possible.

In the background (on the server) the real magic happens but the main differences between using WUMoney and Tapjoy on the Unity side of things is in how you fetch, spend and give Tapjoy currency.

I have named the various callbacks in the WUMoney namespace the same as they are within the Tapjoy namespace so when you follow along with the instructions on the Tapjoy website it should be easy to make the transition to using WUMoney without getting lost or confused. There are only three changes to take note of:

1. You need to use the WUMoney namespace instead of the Tapjoy namespace
2. The delegates have a new format. All of them now take an mbsEvent as the sole argument
3. The function you call have 2 optional additional parameters
    1. **currency** : Specify if you want to work with Tapjoy points or your own custom currency(s)
    2. **meta**: Send a string to the server and have the server send it back. (I.e. product code)

Example:

Tapjoy.OnEarnedCurrency now becomes ***WUMoney.OnEarnedCurrency***
Tapjoy requires the following identity: ***(string currencyName, int amount)*** (and others)
WUMoney always expects the following identity: ***(mbsEvent data)***

Tapjoy uses different function identities for it's various event responders but WUMoney always uses only an mbsEvent object as it's sole argument.

What this means to you is that you can follow along with Tapjoy's instructions on their website and just replace the namespace in front of events to have WUMoney handle the transactions instead of Tapjoy. Then, of course, you need to write the functions that you want to execute when the event fires.

WUMoney has 3 functions to learn:

1. *static public void GetCurrencyBalance(string currency = "points")*
2. *static public void SpendCurrency(int amt, string currency = "points")*
3. *static public void AwardCurrency(int amt, string currency = "points")*

In order to use the Tapjoy points either specify "points" as the currency or just omit it entirely.

Example:

SpendCurrency(30);

Whenever you spend, give or fetch the currency balance, all three functions will always return the current balance of the currency so you can be sure to keep your game up to date.
This can be quite useful in case someone signed in on multiple devices and is playing multiple copies of the game at once. WUMoney won't allow you to spend what you don't have but if you tried to then the server response is a good way to get the correct balance and keep the various device's balances up to date and synced with the real values from the server.

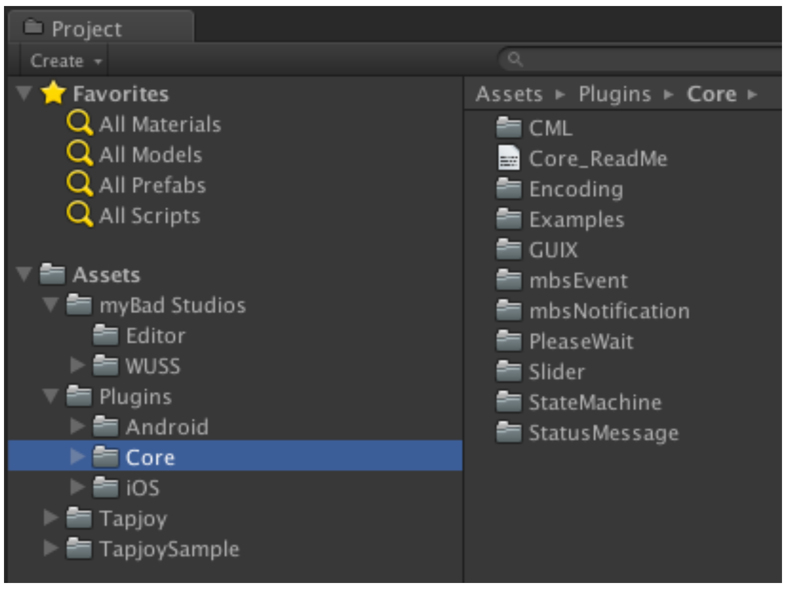
Using the exact same functions as above, but specifying a different value for the currency, will create and use a custom virtual currency for your game.
Creating custom currencies for your game is that easy. Absolutely 0 configuration required at all.

Steps to take before you get started in Unity:

1. Create an account on Tapjoy.com ***(if you haven't done so already)***
2. Follow the instructions on Tapjoy.com to create a game ***(if you haven't done so already)***
3. Follow the instructions on Tapjoy.com to create a non managed currency for iOS and Android for that game ***(if you haven't done so already)***
4. Go to your Wordpress Dashboard and install the wuss_login and wuss_money plugins ***(if you haven't done so already)***
5. Click on the AllGames submenu under WUSS and create a game ***(if you haven't done so already)***
6. Click on the WUSS menu and then select the Money tab
7. Select your game from the dropdown list
8. Enter your iOS and Android currency keys from Tapjoy into their respective fields and click on the Update button next to each one
9. Click on WUSS, select your game in the drop down list and remember the Game ID for later

Steps to get the demo up and running:

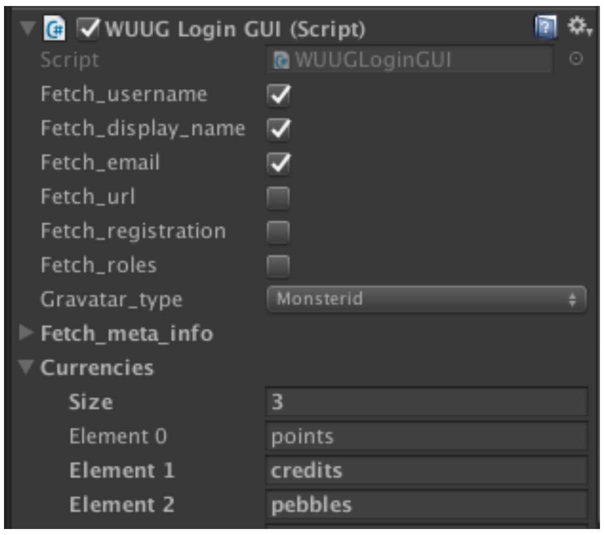
1. Install MBSCore and WULogin for MBSCore (or just WULogin Standalone)
2. Install Tapjoy SDK v11.10.0 (available for free from Tapjoy.com)
3. Under Plugins/Android rename TapjoySampleAndroidManifest to AndroidManifest
4. Move myBadStudios/Core to Plugins/ so Tapjoy can use MBSCore's classes



```
Project
Create
Favorites
    All Materials
    All Models
    All Prefabs
    All Scripts
Assets
  myBad Studios
    Editor
    WUSS
  Plugins
    Android
    Core
    iOS
  Tapjoy
  TapjoySample
```

```
Assets ▸ Plugins ▸ Core ▸
CML
Core_ReadMe
Encoding
Examples
GUIX
mbsEvent
mbsNotification
PleaseWait
Slider
StateMachine
StatusMessage
```

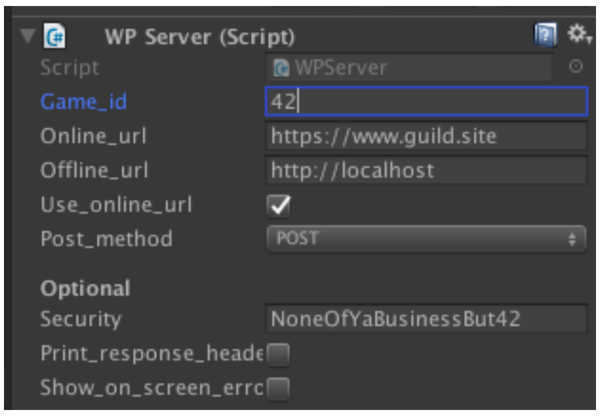
5. Add the two scenes in myBadStudios/WUSS/Demo/Money/ to the build settings making sure to set MoneyDemo_Bootstrap to load first
6. Add your website URL to the login prefab in myBadStudios/WUSS/Prefabs/WULoginUGUI
7. If you want to automatically fetch your virtual currency balance when you login (which we do), enter the names of all currencies your game uses into the 'Currencies" field of the WULoginUGUI prefab. For the included demo scene that means making 3 entries: points, credits and pebbles. If you omit points, it will be added automatically since points is what the Tapjoy currency is called



WUUG Login GUI (Script)

Script	WUUGLoginGUI
Fetch_username	✓
Fetch_display_name	✓
Fetch_email	✓
Fetch_url	
Fetch_registration	
Fetch_roles	
Gravatar_type	Monsterid
Fetch_meta_info	
Currencies	
Size	3
Element 0	points
Element 1	credits
Element 2	pebbles

8. **VERY IMPORTANT (!!!!!!)** Set your Game_id to the value you got from step 9 in the instructions above



WP Server (Script)

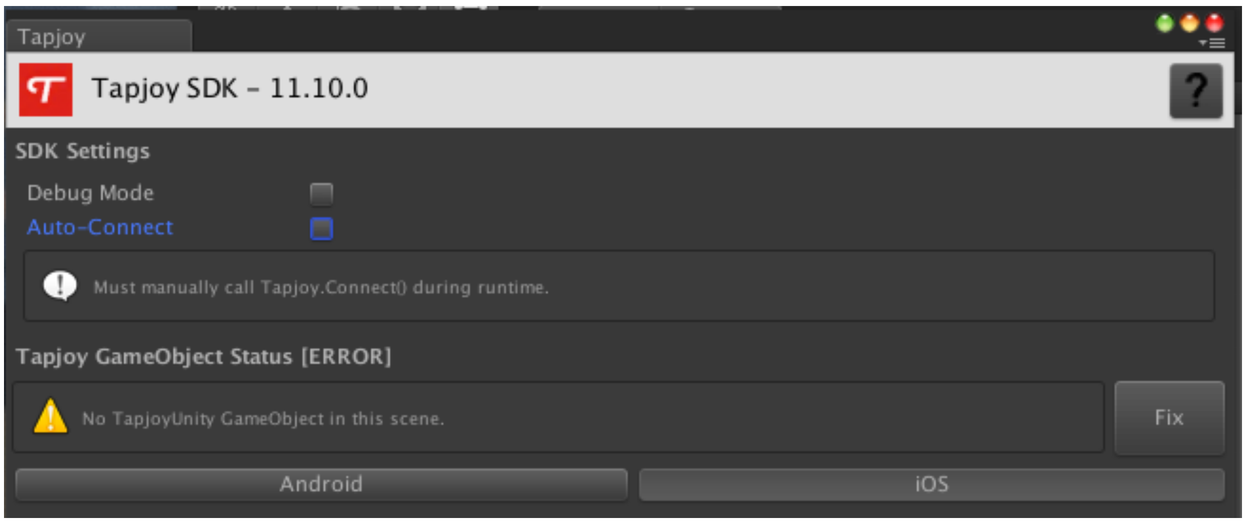
Script	WPServer
Game_id	42
Online_url	https://www.guild.site
Offline_url	http://localhost
Use_online_url	✓
Post_method	POST
Optional	
Security	NoneOfYaBusinessBut42
Print_response_heade	
Show_on_screen_errc	

9. Switch your build platform to IOS or Android
10. Open the myBadStudios/WUSS/Demo/Money/MoneyDemo_Bootstrap scene
11. Configure the WUTapjoyBootstrap component on the TapjoyUnity GameObject
    1. If you already have something else in the scene that handles loading the next scene then simply set Load_new_scene to None. If you plan to have this object and the WULogin kit in the same scene then select AfterIDWasSet. Alternatively, set it to "Immediately" to load the next scene during Start.
    2. Next_scene is the name of the scene to load, if any
    3. Enter your iOS and Android SDK keys
       Also enter them into the Windows->Tapjoy control panel to get rid of the warning messages in there
    4. Turn on "Show_errors_on_screen" during development but turn it off before going live

12. Open the Tapjoy configuration screen by clicking on Windows->Tapjoy
13. Turn off "Auto-connect" and click the "Fix" button

Steps to get up and running in your own project:

1. Follow steps 1 to 4 from above
2. If you don't have a bootstrap scene, create one. This scene is one that loads when the game starts up and is then never loaded ever again. The purpose of this scene is to load the Tapjoy system so you can use it later.
3. Click on Windows->Tapjoy and it will warn you that there is no TapjoyUnity GameObject in the scene. Simply click the Fix button to create the object


Tapjoy

Tapjoy SDK - 11.10.0

SDK Settings

Debug Mode

Auto-Connect

Must manually call Tapjoy.Connect() during runtime.

Tapjoy GameObject Status [ERROR]

No TapjoyUnity GameObject in this scene.

4. Add the WUTapjoyBootstrap component to that same object and configure it like step 11 above
5. Optionally add a WUTJPlacement components for every placement you want to handle (if any)
6. You can now either create a new scene (recommended) and continue in there or continue in the same scene (like i do in the demo. This is almost always going to be a bad idea…)
7. Create a canvas and drop in the WULogin prefab.
8. Add a WUDontDestroyOnLoad component to the canvas you placed the WULoginUGUI prefab under
9. Follow along from step 6 in the instructions above.
   10 is not applicable and you already did step 11 so you can skip those two.

Optional steps:

On Tapjoy.com you will find instructions on how to use the Tapjoy SDK to make use of placements. You can also see this in action by looking at the Assets/TapjoySample sample project that is installed with the Tapjoy SDK.
You are more than welcome to follow these instructions and do it like they instruct but there is an optional alternate way that is a lot simpler, faster and requires a lot less typing…

The "right way" of doing things is that you go to your Tapjoy dashboard, create placements and call them whatever you want then in Unity, when you use a placement, you pass the placement's name to the function as a string.
You then also declare a bunch of event handlers for each placement and write the appropriate functions that each of these events have to execute.

Included with WUMoney is a script called WUTJPlacement. In stead of doing all the legwork that they describe on the website, all you need to do is drop one of these components onto the TapjoyUnity object in your Bootstrap scene and select a placement from the dropdown list. Done. I take care of the rest. All that is left for you to do is to call the static ***WUTJPlacements.ShowPlacement(placement)*** function, passing the placement you want to use via the eTJPlacements enum whenever you want to make use of the placement. That's it. 1 line of code and i do the rest.

Of course, there is one catch… Anything that makes life THAT easy always comes at a price. Fortunately, all I charge you is your freedom of choice, nothing more. Let me explain…

Before, you had the option of calling your placements whatever you wanted to call them but in exchange for the simplicity that WUTJPlacement offers you, it requires that you name your placements the same as the enum values.

Tapjoy has recommended a bunch of placements that you should use in your games and those recommended placements are what I use in the enum. You will see that some of them look funny but I use them exactly as Tapjoy recommended them on their website. Note that when you create your placement names, replace the underscore with a space. They used spaces in their placement names but I can't use spaces in an enum value definition so that is the reason for this discrepancy.

For example:
   if you selected ***Out_Of_Energy*** from the drop down then name your placement ***Out Of Energy*** in the Tapjoy dashboard.

So, technically, using the WUTJPlacement forces you to use the placement names that Tapjoy recommends and forces you to more closely follow their recommended practices. This freedom of choice I take away from you is actually forcing you to add better compliance with Tapjoy's recommended usage.

The placements names I specify in the enum are:

Logged_In	App_Resume	App_Close

Main_Menu	Pause	Stage_Open
Stage_Complete	Stage_Failed	Level_Up
Store_Open	In_App_Purchase	Abandon_In_App_Purchase
Sign_Up_Complete	User_High_Score	Virtual_Good_Purchased
Out_Of_Goods	Out_Of_Energy	Finished_Tutorial
OfferWall_Request	Interstitial_Request	Video_Request

The bottom three are generic / general use ones I added to the list to give you back some freedom :D