How to use DLC In Blueprints Plug-in for Unreal Engine with EGS

If the plugin works well for you please consider leaving a review letting us know what you like about it.

Notes:

1. This product is not made by Valve/Steam/Epic Games, it is a custom plugin not associated with them.
2. In order to test the functionality with the epic games store such as checkout or dlc ownership, the game will need to be launched from the epic launcher. Currently Epic games store does not support testing in the editor itself though this could change in future engine versions.

3. Before you add step 7 to your project, please set up your EOSOnlineSubsystem stuff in the project settings. There is a bug in the unreal engine that will crash on startup if that step is reversed in order.
4. If you are not getting the features to work in live sandbox on EGS then make sure you go to product settings-> Sandboxes-> Live->Deployments and assign the live sandbox to it, if you don't it will work on stage but not live.
5. Please ensure your client policy on epic games store supports ECommerce features if you are using this plugin with DLC features.

Getting Setup:

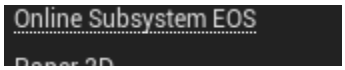
Step 1. Install plugin via unreal marketplace/epic games launcher.

Step 2. Enable plugin in unreal engine project plugin manager.

Step 3.Verify that the “Online Subsystem EOS” plugin is enabled in the unreal engine project plugin manager. This should auto enable when you install the plugin but it is best to double check.

Step 4. Restart engine/project to apply plugin changes to project/engine.

Step 5. Go to your project settings and under “plugins” setup your “OnlineSubsystemEOS” stuff



Step 6. Scroll Down to the very bottom of the docs to see what needs to be in the “OnlineSubsystemEOS” settings page.

Step 7. Head to your “defaultengine.ini” file under “projectdirectory/Config” Folder and copy paste this into it. If you want to use it with steam and epic at the same time then follow the documentation for “EOS Plus” here
https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Online/EOS/

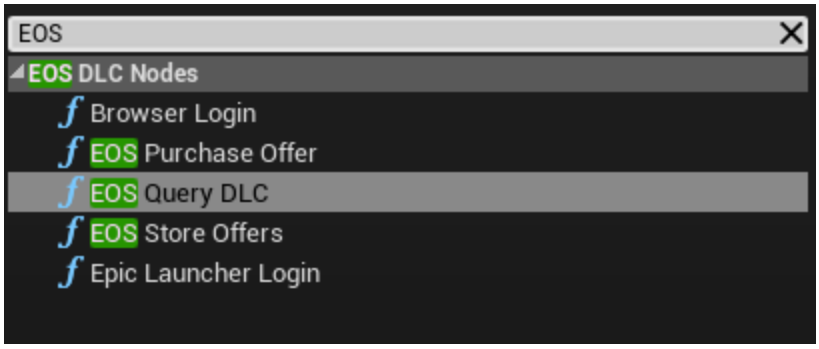
“[OnlineSubsystemEOS]
bEnabled=**true**

[OnlineSubsystem]
DefaultPlatformService=EOS

[/Script/Engine.GameEngine]
+NetDriverDefinitions=(DefName="GameNetDriver",DriverClassName="OnlineSubsystemEOS.NetDriverEOS",DriverClassNameFallback="OnlineSubsystemUtils.IpNetDriver")

[/Script/OnlineSubsystemEOS.NetDriverEOS]
bIsUsingP2PSockets=**true**
”

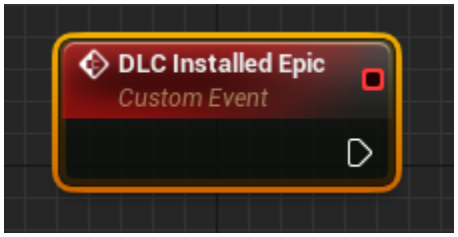
How to access the plugin in Blueprints:
Step 1. Right click in the blueprints to open the context-sensitive menu for nodes.
Step 2. Search for the keyword “Get EOS_Subsystem”.
Step 3. Drag off of the return value for “EOS_Subsystem” and search “EOS” ; all nodes should be under the “EOS DLC Nodes” category.



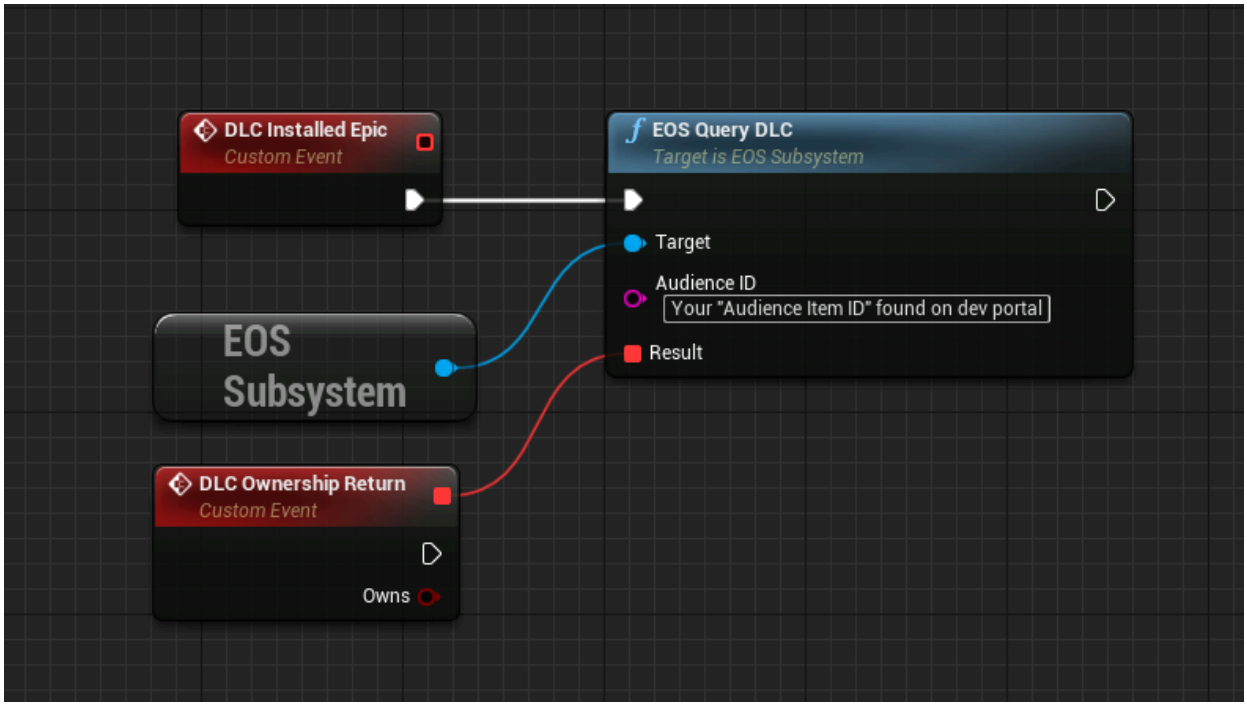
All features/nodes that come with the plugin:

<u>EOS Query DLC:</u>
This node works by checking if the player on epic games store has purchased the DLC on their EGS account(all of that is handled by Epic). It will return a custom event with a bool output that can be used to set variables, branch checks, etc. for DLC ownership logic.

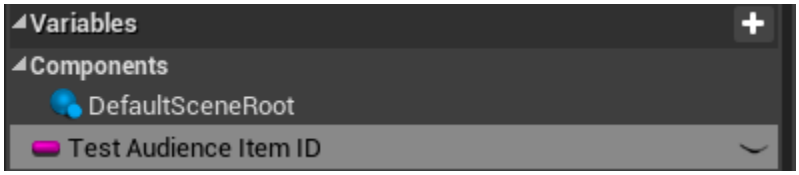
Example on how to use “EOS Query DLC”:
Step 1. Create a custom event that can be used to call the query DLC function anywhere you want in your event graph.



Step 2. Use the “Get EOS_Subsystem” node to call the “EOS Query DLC” function, this function will return a custom event on the “Result” input, all you need to do is drag off of the “result” and click “create custom event” and it will make one for you with the “owns” bool return.



Step 3. Find your “Audience Item ID” for your offer on the epic games store dev portal, once you have that store it as a variable for easy use. Each DLC will have its own unique Audience Item ID so you can use this with as many DLC as you want.
Note: This needs to be a string variable type
Note: This also can be used to verify ownership of the full game, not just DLC.

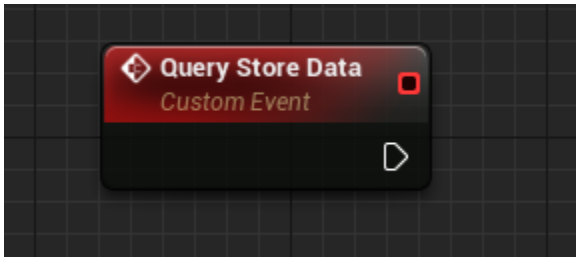


Step 4: Assign the Audience Item ID variable to the node input and any logic you want to do after ownership has been verified will go on the “owns” return boolean.

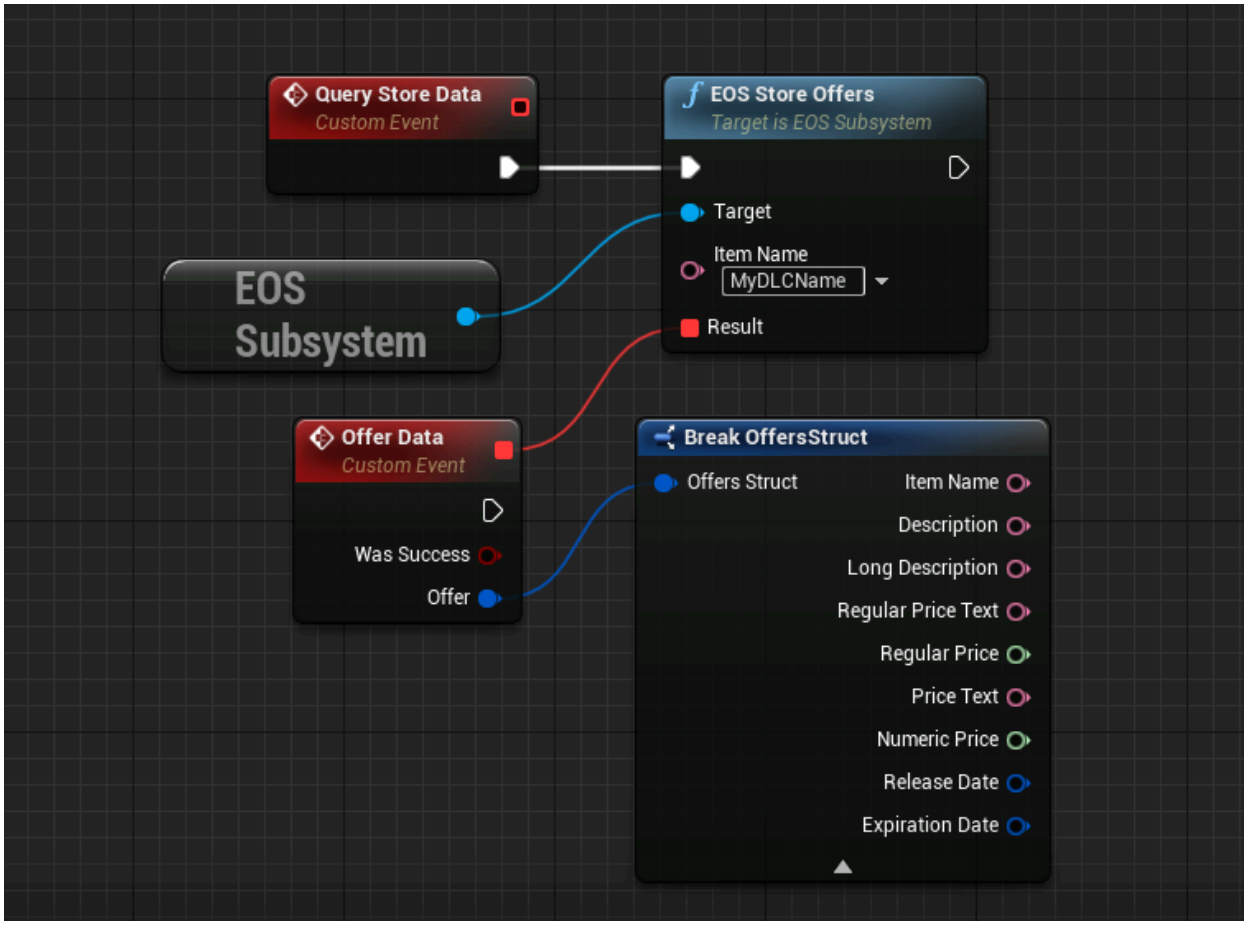
Best Practices: I recommend using this node on the “login”(See login) node success value and then just store the “Owns” as a bool somewhere like a game instance, if someone purchases it(see “checkout”) then just update that bool.

Example on how to use the “EOS Store Offers”:

Step 1. Create another custom event, this event will get the data for an offer on Epic Games Store and return its data such as its current price, name, description, etc.



Step 2. Pull off of the “Get EOS_Subsystem” node and call “EOS Store Offers” function. This will take two parameters. 1. An item name which is the literal name of your DLC offer in the dev portal and 2. A custom event return, for the custom event drag off of “result” and click “Create custom event” and it will automatically be created for you with the proper return values.



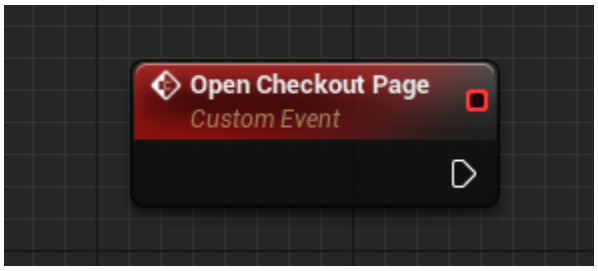
Step 3. To get the offer data, break the “Offer” struct with either “break struct” node or by right clicking the output and clicking “split struct pin.

Step 4. From there use any logic you want such as binding text to a price, store widgets, title names, etc.

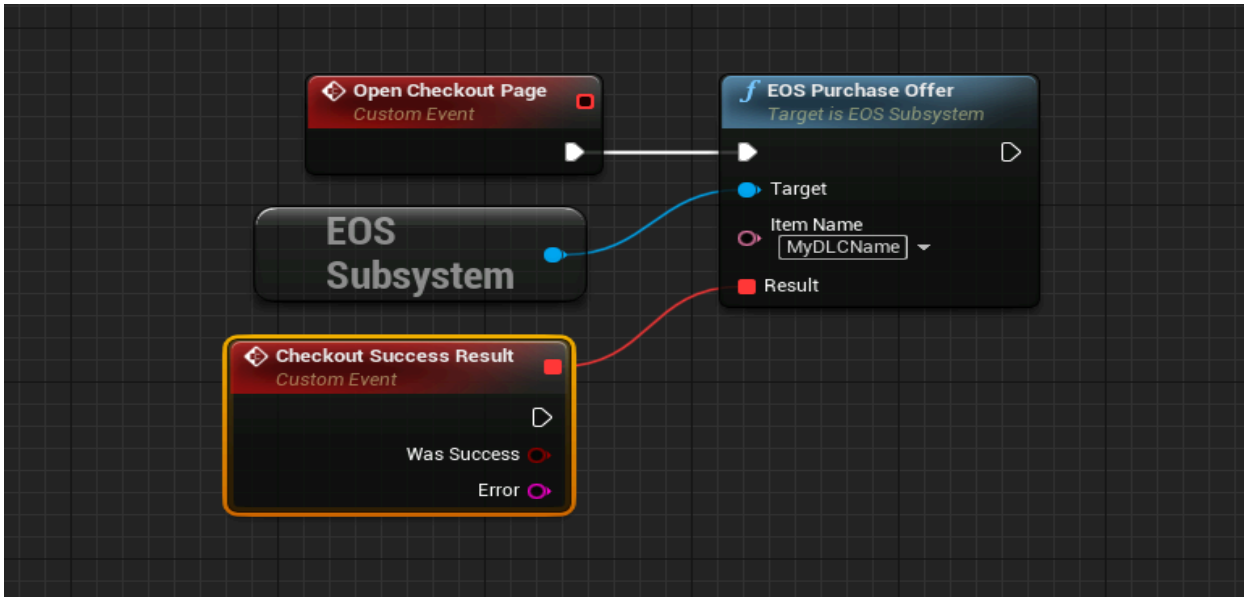
Note: The price does not contain decimals or currency symbols like $, you will need to use the “append” node to add those symbols in for styling.

Example on how to use the “EOS Purchase Offer”:
This node will open a direct checkout screen from within the game via the epic games launcher. You see this in games like fortnite where when you click purchase the checkout screen automatically pops up in game.

Step 1. Create a new custom event that will hold a function for opening the checkout overlay window for purchasing in game.



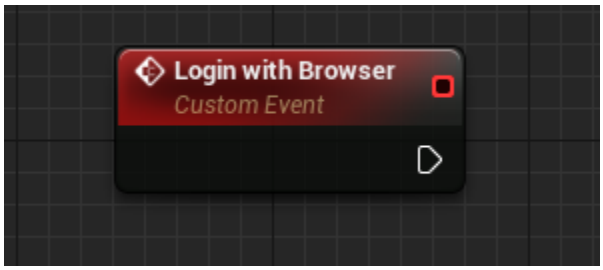
Step 2. Add the “EOS Purchase Offer node to the custom event from the “Get EOS_Subsystem” node. Once you have done that pull off of the “result” input and click “add custom event” in the context menu.



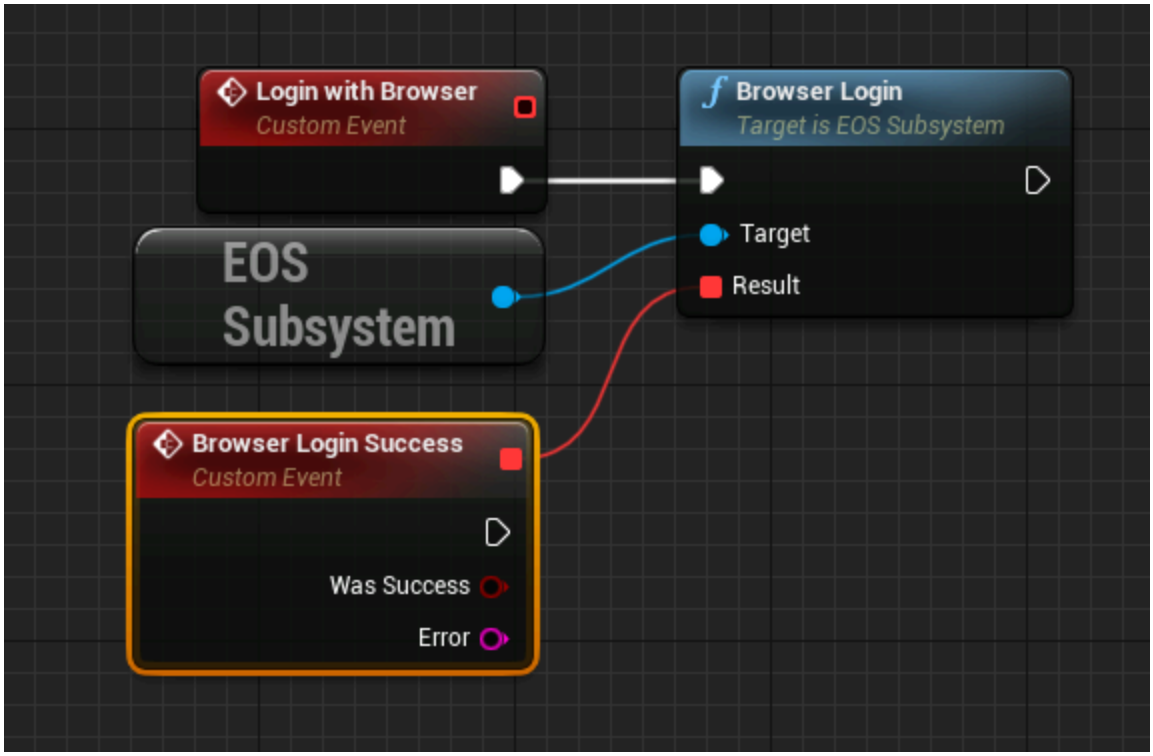
Step 3. Type in your DLC Name to the “Item Name” input or create a text type variable and put the DLC name in there. Earlier in the documentation we had set a bool on “Query DLC” node on init or begin play, if the checkout “was success” you should update that bool here with a branch check, this way any logic that checks for DLC ownership from that bool will be updated.

Example on how to use the “Browser Login”:
This node will open a window to login the user via a web browser, this is usually used for testing but can be used in a commercial game for something like cross platform launches. If you are focusing on epic games store publishing then I recommend the “Login with launcher” as it is automatic.

Step 1. Create a new custom event that will hold a function for opening the login web portal function.



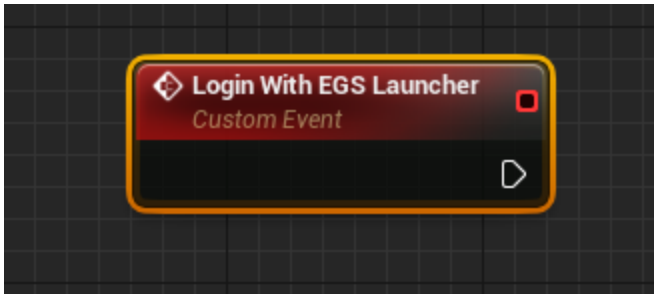
Step 2. On the “Get EOS_Subsystem” node return call the “browser login function” and on the “result” input drag off and click “Add custom event”



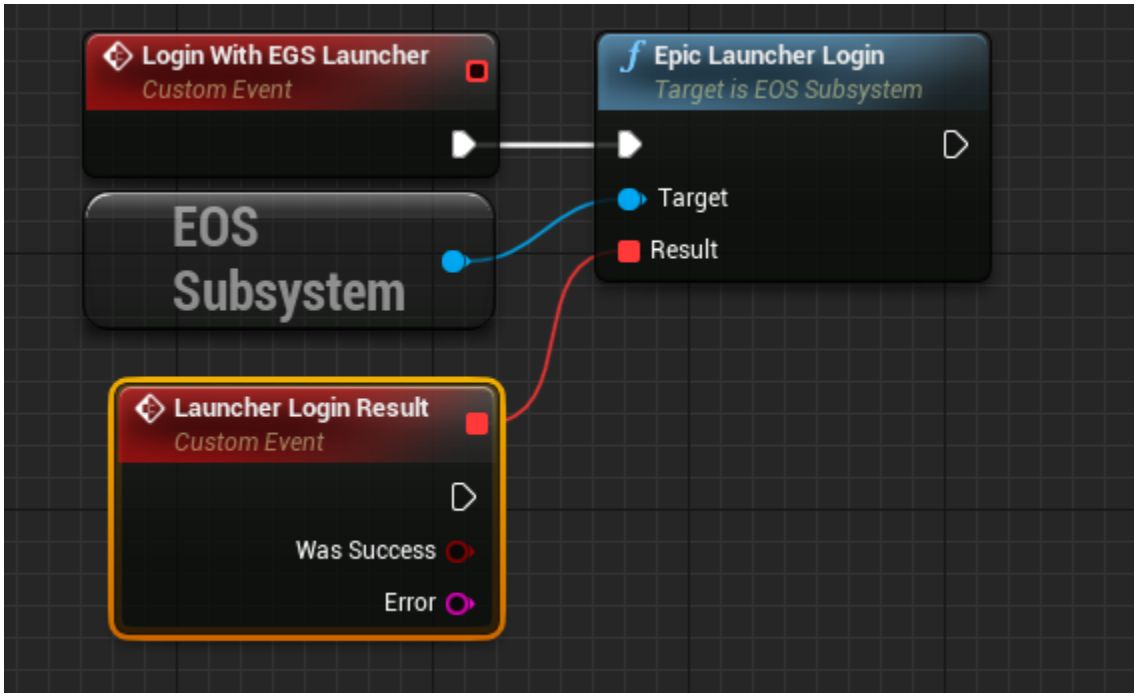
Step 3. It can take a second or two for the login to finalize so I recommend putting anything like DLC ownership checks on the “was success”, if the other nodes run before “was success” returns true then they won't work as they rely on the user being logged in.

Example on how to use the “Epic Launcher Login”:
This node will automatically login a user from the epic games launcher, all that is required is the game is launched from there, if you are doing cross platform and need epic support on those other platforms then I recommend using the “Login with browser” node seen above.

Step 1. Create a new custom event that will hold a function for opening the “Epic Launcher Login” function.



Step 2. On the “Get EOS_Subsystem” node return call the “Epic Launcher Login” and on the “result” input drag off and click “Add custom event”



Step 3. It can take a second or two for the login to finalize so I recommend putting anything like DLC ownership checks on the “was success”, if the other nodes run before “was success” returns true then they won't work as they rely on the user being logged in.

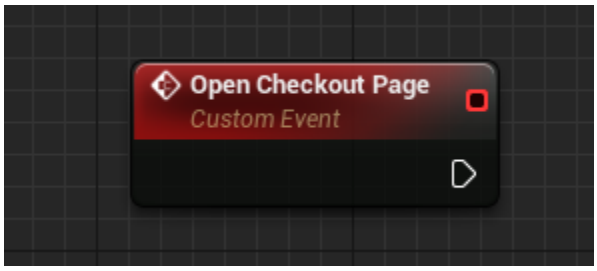
Notes:

1. It is recommended you put this on Begin Play or event init of a game instance since the login credentials from the launcher only lasts in the game for about 2-3 seconds before it goes away. The way epic games handle it in their games is on a widget construct that opens up right after event init.
2. If this is being used on begin play or event init, then I recommend putting a delay node on there for like 0.2 seconds as event init and event begin play can execute too

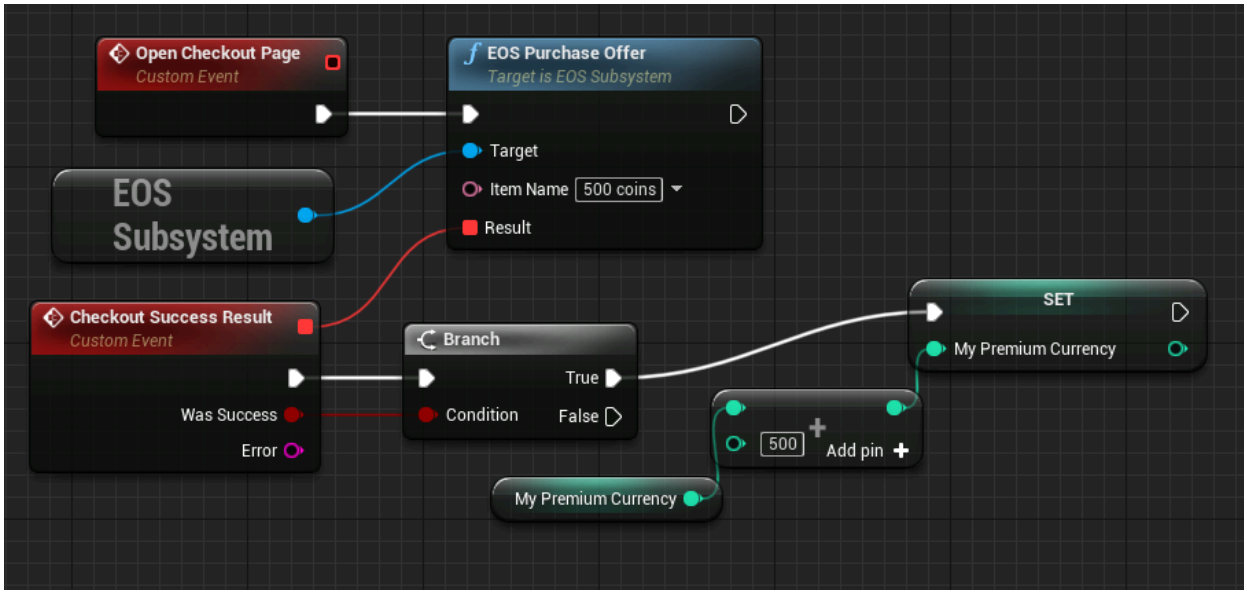
quickly sometimes for the code to run properly.

Example on how to use the “EOS Purchase Offer” for micro-transactions(epic games store only):
This node will open a direct checkout screen from within the game via the epic games launcher. You see this in games like fortnite where when you click purchase the checkout screen automatically pops up in game.

Step 1. Create a new custom event that will hold a function for opening the checkout overlay window for purchasing in game.



Step 2. Add the “EOS Purchase Offer” node to the custom event from the “Get EOS_Subsystem” node. Once you have done that, pull off of the “result” input and click “add custom event” in the context menu, this will automatically create a custom event with the proper return data.

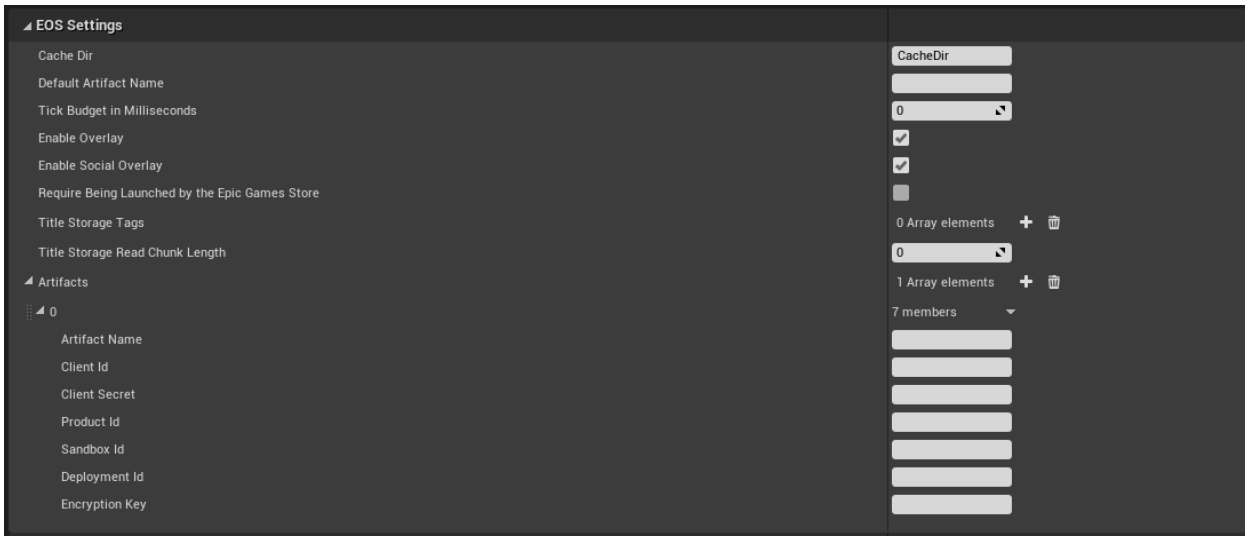


Step 3. Type in your consumable purchase name to the “Item Name” input or create a text type variable and put the consumable purchase name in there. On the “was success” bool result increment the variable you use for currency with a branch check.

How to setup project settings for OnlineSubsystemEOS project settings to work with epic games store(From step 6):

When you open this window in the project settings you will see this screen, I will go through what each one needs to be and where to find them.

I recommend creating 3 artifacts, one for each sandbox (Dev, Stage, Live), if you sign up for EGS they will tell you what those are.



Artifact Name: The string associated with your artifact ID..
Client ID: This is found on your “product settings” on the EGS dev portal, just copy paste this into the field. It will remain the same for all sandboxes.
Client Secret: This is next to the Client ID on the dev portal, copy paste this in as well.
Product ID: This is found in the Dev portal as well under “product settings”, copy paste this in. it will work with all sandboxes.
Sandbox ID: This will be different for each sandbox you have(dev, stage, live), I recommend making 3 artifacts each with one of these representing them.
Deployment ID: This is directly associated with the Sandbox ID, you can find it right under the sandbox ID page on the dev portal.
Encryption key: This is a 64 length string that you need to generate, unreal engine uses this string to generate encryption between the server, it is required for this to work.

I recommend this site to generate the string, just choose “64 digits” and copy paste one of the generated results, please make them different for each artifact.
https://www.browserling.com/tools/random-hex

Notes:
1.In order for the checkout window and EGS overlay to open you need to click the checkboxes for the respective ones in the project settings. Both enable social overlay and enable overlay are required for the checkout stuff.
2. The checkout and EGS overlay will not work in the editor, epic may update the engine to support it later but as of right now the game needs to be launched from the EGS in stage build for the overlay and checkout/DLC stuff to work. The epic launcher runs command line arguments that the plugin code receives to work, which is why the editor does not support it yet.