Waiting for Data

Due to the nature of network delays, and the process of fetching data from the blockchain. The data will not be available for a short period. This can be anywhere from a few frames to a few minutes.
Defungify pulls data on request. Once the data is fetched, we cache the data for 24 hours. Subsequent pulls should take an insignificant, but non-zero, amount of time.
In any case, the data will not be available for at least a few frames.

There are a few options on how to handle this problem. But first, we will describe the general process.

Here is the list of Blueprint nodes that will launch an API request. Repeatedly calling these blueprint nodes will result in multiple requests. This will likely result in a request that will never be completed.
-Token In Wallet
-By Token ID
-By User ID
-By Wallet
-Start Validate
-Validate Wallet Token ID
-Check User ID

Each of these will return an object. This object will be used as an input in future functions. Be sure to store this object as a variable.

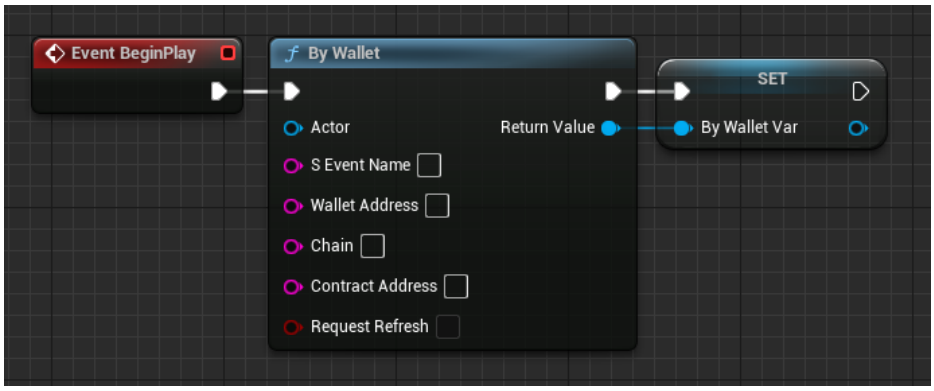
You can use the “Get Status Validate”/“Get Status” to get information regarding the current status.
Is Done – Is the request complete?
Was Success – Was the request successful?
Message – If there was an error, the error message will be here. If the call was successful, there may be some information.
From Cache – Only applies to getting NFT data. If there is a problem reaching the blockchain, or for some reason the updated NFT data is not accessible, this will return true and the data returned will be from the last cached state. (If available)

With this information, the general process is as follows.
1. Request the data
2. Store the resulting variable
3. Wait for the request to complete
4. Fetch the data using the stored variable

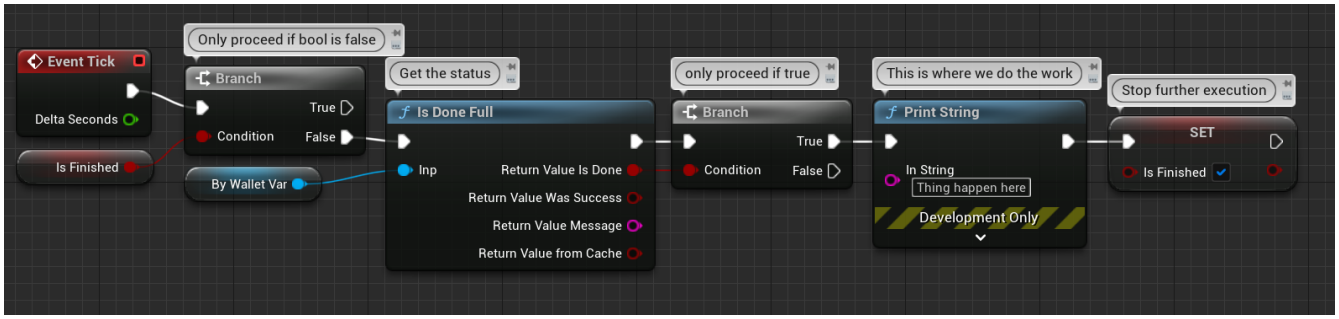
Example Methods

The Lazy Method

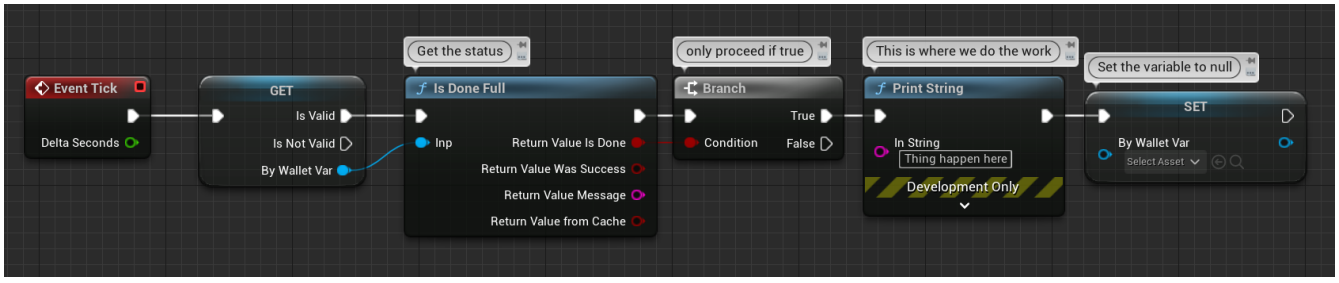
If you're looking for a simple method, or just a temporary solution. You can use a boolean to ensure the data is only executed once.



Starting the request on event BeginPlay ensures only a single request occurs.



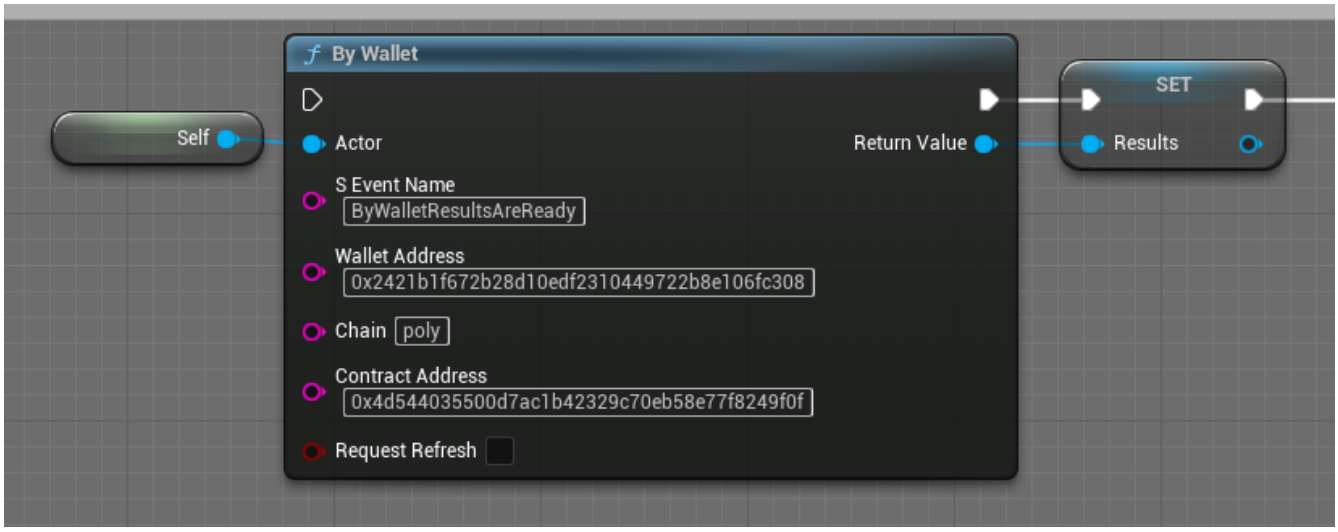
Here is a simple example



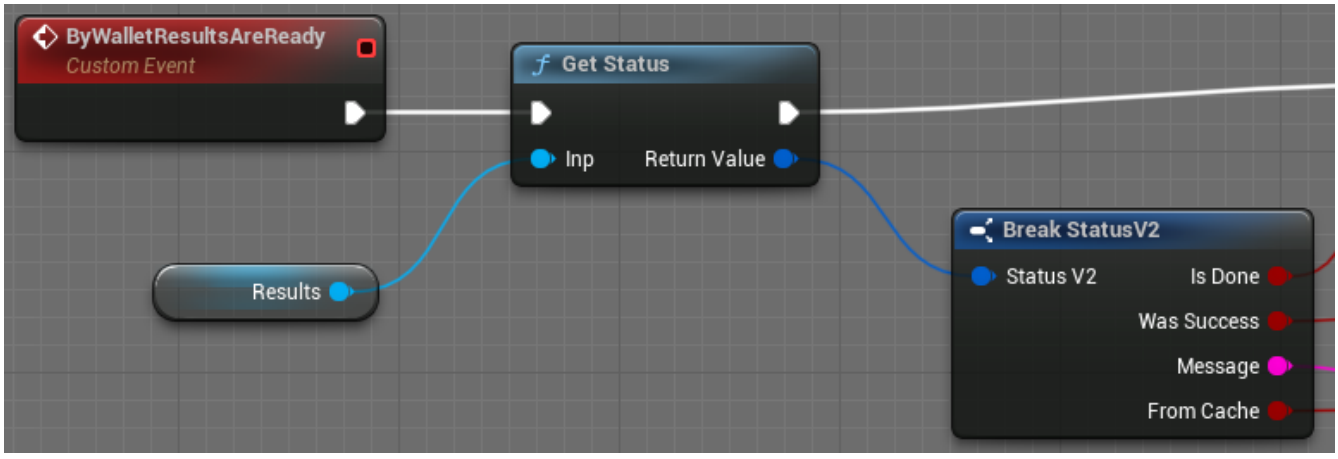
Alternatively, you can instead use a validated get and set the variable to null when finished. (You can convert a get to a validated get by right clicking a get and click "Convert to Validated Get")

Custom Events

We have recently implemented custom events on actors. If your object is an actor, or potentially has access to an actor, you can fire a custom event on the actor.

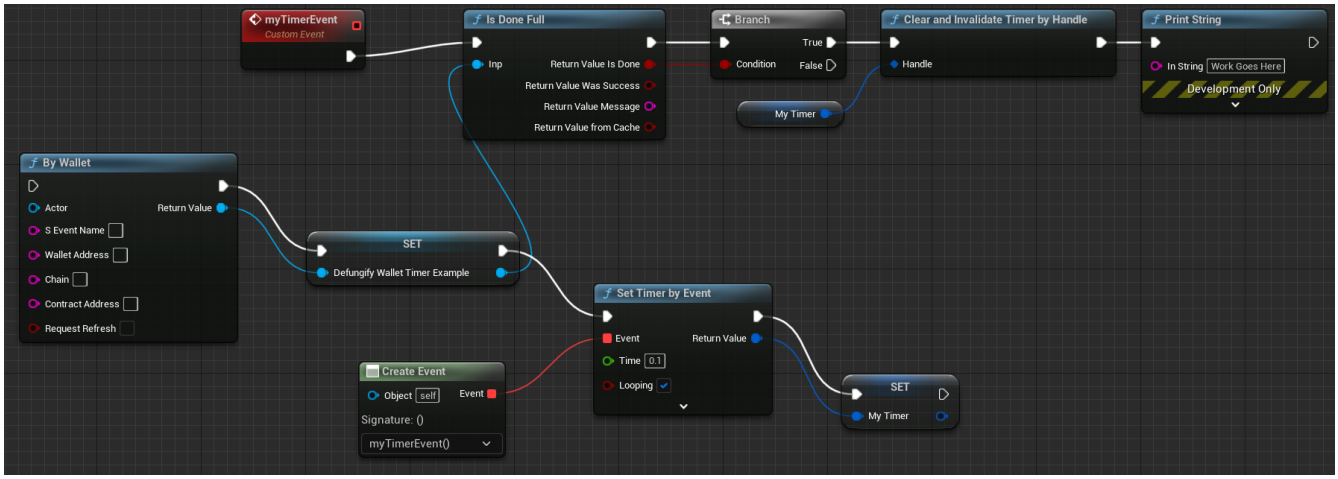


Passing in a reference to the actor (In this example, a reference to self) will provide a target to fire a custom event. In this example, we will be firing the event “ByWalletResultsAreReady” on self.
(To get a reference to self, drag off from Actor and search for “self”.)
Once the request is complete, the custom event specified will be executed.



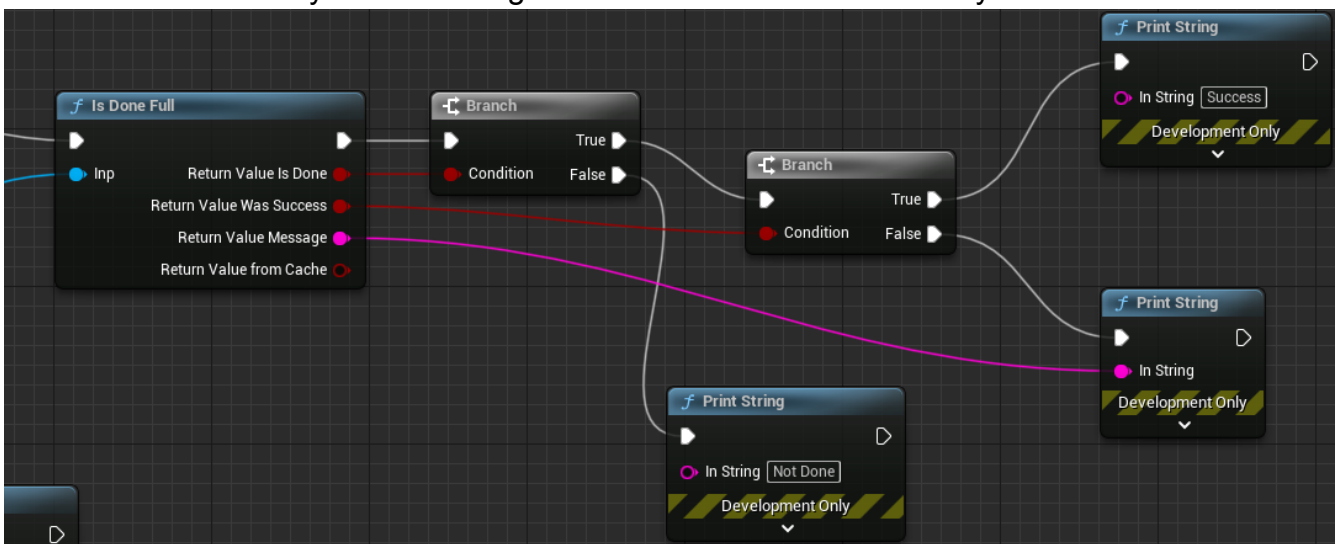
When the custom event fires, you will be able to access the data as normal.

Timer Method



At first glance this may seem like a mess, so let's walk through this.

1. Right click on empty space and search for "Set timer by event". Promote the return value to a variable. We're going to call it "My Timer".
2. Drag off from the Event input and search for "Create Event"
3. Either create an event or click the dropdown and select "create a matching event" and give it a name.
4. Right click on empty space and create the byWallet node (or the method you're trying to use)
5. Drag off the return value and promote it to a variable. Make sure the execution then continues to "Set Timer By Event".
6. Go to the custom event you created. Right click empty space, look for "Is Done Full" / "Is Done Validate" (depending on if you're looking for NFT data or doing a validation task respectively.)
7. Make sure to feed it in the variable you promoted earlier. (From the Defungify node, not the timer)
8. On the Is Done Full, right click the return value and click "Split Struct Pin".
9. There are a few ways to do the logic. This is the recommended way.



The print statements are for debugging purposes.
10. Right click on the empty space and search for “Clear and Invalidate Timer by Handle”. The input here should be the timer variable promoted earlier. The goal here is to stop the timer from continuing to run.

This should be all you need. After the success path, you can continue as normal. See the Pulling NFTs tutorial for more information on that.

The general idea here is that you start the request, then you start a timer that runs every x seconds (0.1 seconds in this example). Each time the timer triggers, the custom event fires. If the request is not finished yet, the execution stops. When we invalidate the timer, the custom event stops firing.