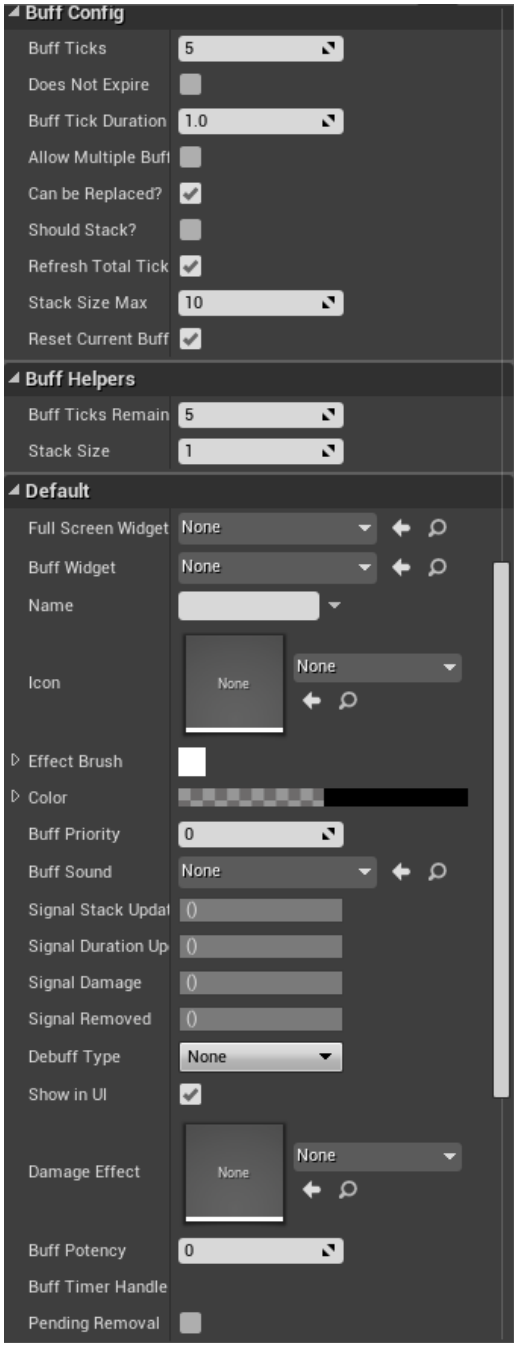
Table of Contents

About this tutorial

This tutorial will go through the basics of creating "buffs". Buffs are effects that last for a short period of time that may grant bonuses or penalties to players or NPCs. We will go through the basics of how to create buffs, (some of the) effects of buffs and also some important pitfalls that needs to be avoided.

Buff datastructure

Buff config



Buff ticks determines how many times the buff should apply the effects in the "Event Gameplay Effects On Tick" event. If your buff is a healing over time or damage over time buff, you won't need to implement custom functions for these as the existing parent blueprints can already handle those freely.

Does not expire - some buffs don't end by a simply reaching the end of it's duration. Do note that if you tick this box, buffs are still removed from the player on logging out from the game.

Buff Tick Duration determines the length of each tick. You may want a buff that applies effects every 1 second or every 0.5 seconds or maybe even every 60 seconds.

Allow Multiple Buff Instances - normally, adding a buff to a player simply results in the stacksize of the buff increasing. However - with this ticked, each buff applied to the player is a separate entity. Typically, you won't need this unless you have a specific effect on a buff that a stacking modifier wouldn't affect.

Can be replaced? needs to be ticked for buffs that, on application, will replace previous instances of itself without stacking.

Should stack? This needs to be ticked if the buff stacks. Stacking means that the effect of the buff is (typically) multiplied by the stack size (for damage buffs, they may also apply an exponent, more about this later in the Poisons section.

Refresh Total Ticks on Stack Increase allows you to set up a buff that resets the amount of ticks once the stack of the buff increases. Most poisons and stacking over-time heals work like this - in essence, the timer of the current buff is reset, while the stack increases.

Stack size max is the maximum amount of stacks the buff may have. Oddly enough.

Reset Current Buff Tick Timer on Stack Increase also does exactly what it says on the box - it resets the timer of the buff back to 0 and starts counting again. It is almost always used in the same buffs as Refresh Total Ticks on Stack Increase.

Buff Helpers

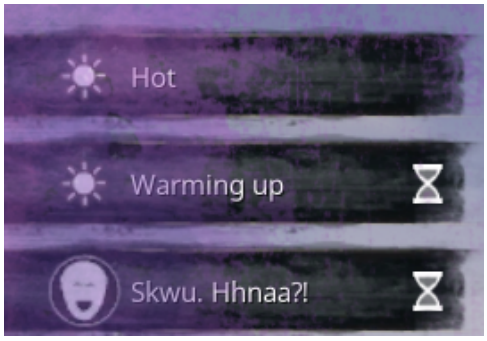
These are simply debug helpers. Typically, you should match the Stack Size Max to **Buff Ticks Remaining** and **Stack Size = 1.**

Default

Custom buffs usually have extra setups beyond the ones listed here, but we will go through those later on in the section that talks about stat buffs, poisons and healing. Each of these types of buffs implement slightly different stats in addition to the default ones that are inherited from the **01_BP_AC_Buff_Master** blueprint.

A number of entries in this section are legacy setups and not used anymore - let's get those over with straight away. They are as follows:

Full Screen Widget	Damage Effect
Buff Widget	Buff timer handle
Buff Priority	Pending Removal



Name is the localized string for the name of the buff. It is the name of the buff that appears on the small little bar on the left hand side of the screen.

Icon is the icon displayed on the buff widget.

Effect Brush is the on-screen overlay for the buff. By expanding this selection, you can select an image and tint for the full-screen effect. Most of our current buffs make use of the same full-screen effect image, which is named **T_BuffFullScreenEffect**

Buff sound links to whatever sound cue you want to play while the buff is running.

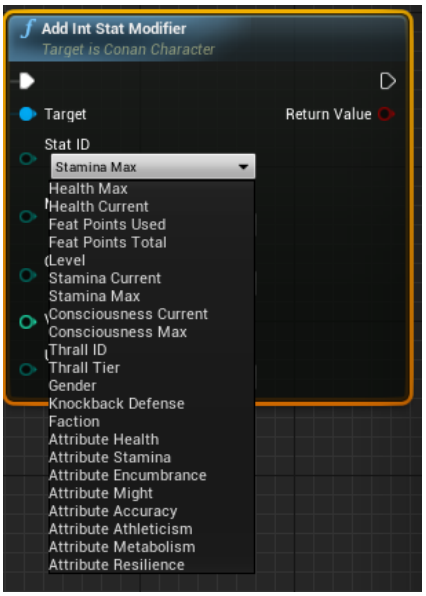
Debuff type you will only really need to touch if you are making a poison effect. While this does not do anything for the buff itself, it helps other buffs and effects identify the buff and cleanse it properly. This entry is a semi-legacy entry in that it has functionality, but really should be moved into the TAGS of the buff.

Show in UI - you may want to not show some buffs in the UI - as an example, if you are using a buff for tracking purposes, you might want to uncheck this. The same goes for most instant-application healing effects.

Buff potency is a very useful setting - you can use this to make buffs automatically override other buffs of the same class if the potency of the buff is higher or lower than the other buff. For example - if you apply BUFF A that heals the player for 10 points over time, and then apply BUFF B that heals the player for 20, you may want to use this functionality. Without it, you will need to end the buff through logic in the Event-graph.

Setting up different types of buffs

Buffs to player stats



Players have multiple stats that can be affected by buffs. The stats are either **FLOAT** stats or **INT** stats.

For an example, if you open the buff "**BP_AC_Buff_MaxStamina**" but if you want to see the list of float stat modifiers instead, use the function "Add Float Stat Modifier".

Float/INT stat modifiers are added to the stat in a safe way. What this means that the original stats are not modified or saved in any way and modifiers do not persist through disconnects/logouts. However, we still need to de-apply the effects of the buff, otherwise the bonus will last until logout.

In the "BP_AC_Buff_MaxStamina" buff, the INT stat "Stamina Max" is modified. As you can see, it adds 10 stamina on buff application and then applies a -10 stamina on buff cleared.

Poisons and damage 'buffs'

When it comes to damage causing buffs, there are a few different ways of implementing them:

00_BP_AC_Buff_Exponential_Master is the parent of all buffs that have damages that do an increasing amount of damage based on the stack size and an exponent of "3". What this means is that if the player receives a single stack, the damage equates to a single stack. But if there are 2 stacks, it treats the buff as if there was "2*2*2" = 8 stacks (an exponent of 3, see?). These are the most lethal of all damage buffs because of the scaling.

00_BP_AC_Buff_PercentDamageMaster is the parent of all buffs that do a percentage of the players health in damage. It's worth knowing that since Unreal doesn't have % input fields, we are using a float stat for this - meaning that the **PercentDamage** stat for these buffs might say "0.01", which means 1% of maximum health. Of course, these buffs also multiply by stack size.

00_BP_AC_Buff_Damage_OverTime_Master is the most straightforward method of doing direct damage over time. The stat DamagePerBuffTick simply determines how much damage the player takes every tick the buff effect triggers.

Unless you want to create custom functionality for damage buffs, you don't need to use the Event-Graph at all. All the stats are readily available in the Details window in Unreal and follows the Data-structure mentioned before. What this means is that as long as you make a new buff as a child of any of the above-mentioned buff-types, all you need to do is to click "Class Defaults"and edit the buff.

Healing buffs

As with damage buffs, healing buffs do not require you to touch the Event-Graph unless you really need to create custom functionality.

00_BP_AC_Buff_Heal_OverTime_Master has a three-in-one function that allows for both creating healing-over-time effects as well as instant heals; all controlled through the Class Defaults.

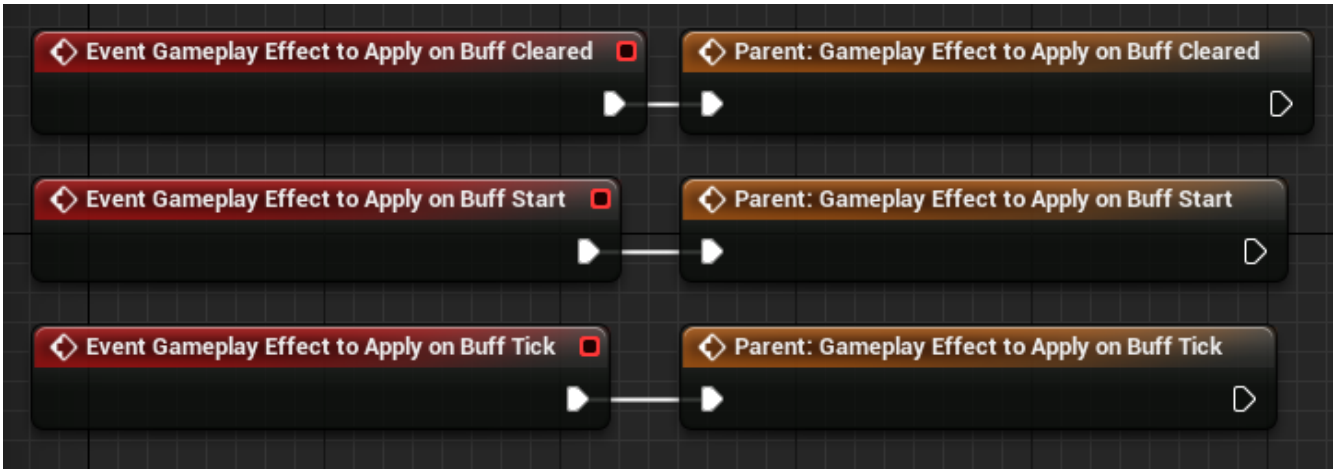
The stat **InitialHealthBoost** determines how much healing the buff should give the instant it begins. If you want to create an instant-heal, this is where you want to put in the healing amount. In addition, you should set the StackSizeMax to 1 and the BuffTickDuration to 0.01 to stop the buff from hanging around (unless you have a specific reason for doing so)

Useful Events

When making custom buffs that have effects on the player beyond the ones listed above, there are some events you should be aware of. Depending on the situation, you may or may not want to also run the parent blueprint functionality or not.

A quick primer for what "parent calls" do might be in order. If you know this, feel free to skip ahead.

If the parent blueprint has functionality in an event and you want to access it (for example, a healing-over-time buff that you also want to implement custom functions in), you will need to implement a parent function-call. In order to do so, right click the event (for example "Event Gameplay Effect to Apply on Buff Tick" and select "Add call to parent function" and link them as in the image below.

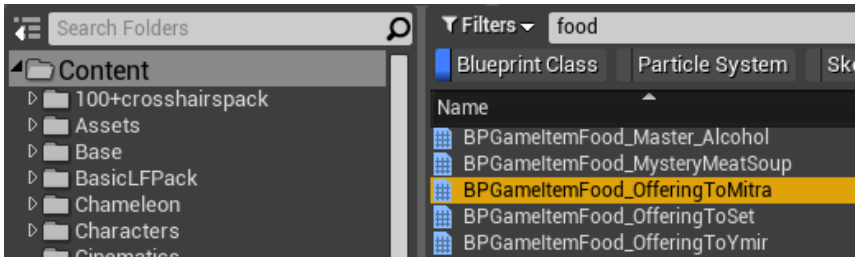


If you don't do this, you will override any functionality of the parent, which means that your buff will work except for the healing part.

There are additional events that may be useful as well, accessible in the Event-graph by right clicking and typing "Event buff" in the filter. For the most part, however, the already mentioned events will be the ones you want to use.

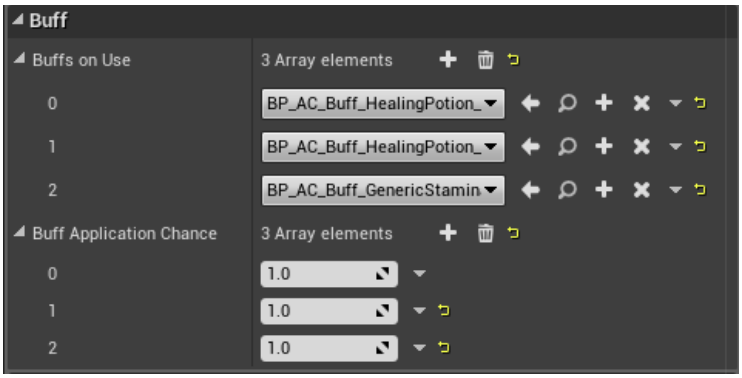
Applying buffs through items

Attaching a buff to a consumable



Consumables such as foods or drinks is the easiest way of applying buffs.

In this guide, we won't go through how to create a food item (although it's very easy - simply copy the stats of one of the existing ones in the ItemTable and create and link it to a new food blueprint - you can use any of the existing ones, for example the **BPGameItemFood_OfferingToMitra** blueprint).



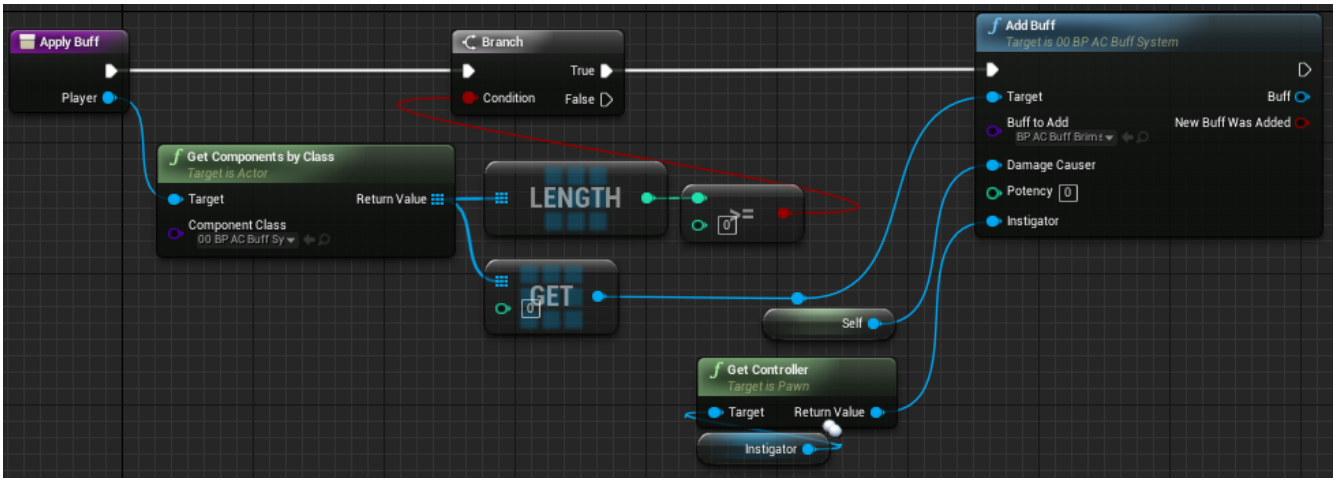
When you open a food blueprint, you can access the class defaults and in the Details window, you will see something similar to the image on the right.

The **Buffs on Use** is a list of the various buffs that will apply to the character when the food is consumed.

The Buff Application Chance is a percentage chance of application, allowing you to create foods that not always apply the buffs. If you want the buffs to always apply, 1 equals 100% (0.01 equals 1%)

Applying buffs from other sources

You may want to apply buffs from other sources, such as an area. An good example blueprint that does this is the BP_HazardArea blueprint but in essence, what you need to do is to get the **00_BP_AC_BuffSystemComponent** from the player and use that to add a buff (see image below).



Applying buffs through the energy system

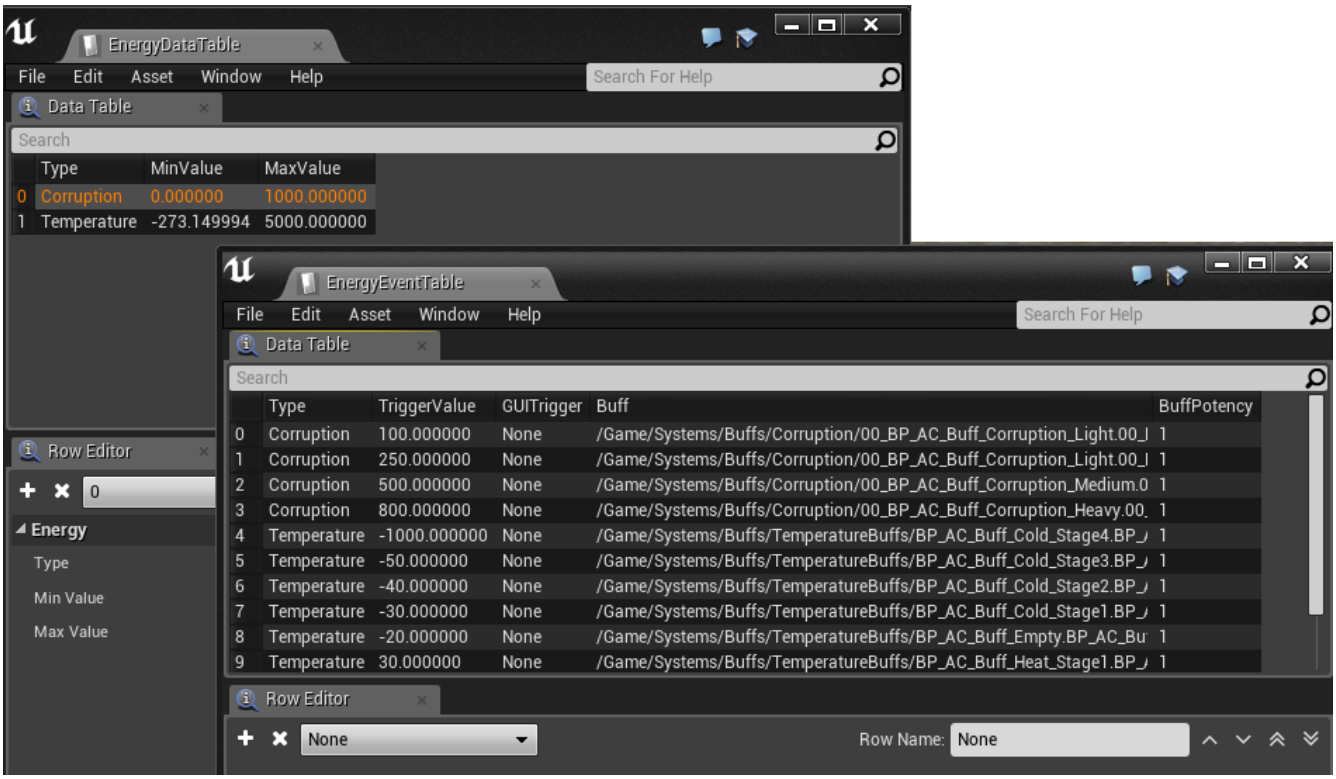
The energy system currently only deals with Corruption and Temperature. It is a way of applying buffs by checking if the player is above or below a certain threshold.

This guide will not go through energy types in detail, but suffice to say that there are two tables - the EnergyDataTable that contains all the different energy types of the game (currently, Temperature and Corruption, as mentioned) - it's possible to add more types to the table, but you will need to implement the application of said energy types yourself.

The other table, where the actual buff application exists, is the EnergyEventTable. This table will check against the energy stored on the player and compare against the table. If the player's internal energy amount exceeds the Triggervalue of the table, the related buff is started on the player.

If the player energy value passes another threshold (or dips down below the current one), the previous buff is automatically removed and another buff is run linked to the new threshold.

Worth noting is that the table is read from the first entry and onwards, which means you will need to implement the triggers in an escalating order.



Pitfalls to be avoided

There are some common mistakes that might not be easy to spot when making buffs and we should go through some of the most common ones. While some have partially been talked about already, we will repeat them here briefly.

Using the ExponentialMaster can be dangerous

Why? Simply because of the power of the exponent. Since the exponent is hard set in the buff master as "3", there is very little wiggle-room for the strength of the buff.

Forgetting to de-apply the effects when the buff ends

This is easily remedied. Make certain that the effects applied to the character are reversed when the buff ends by using the Event Gameplay Effects to Apply on Buff Cleared.

However, there is another pitfall related to this which is much much nastier and deserves to be gone over in detail.

If a player disconnects from the game while a buff is still running, the Event Gameplay Effects to Apply on Buff Cleared is not run. While this is not a big deal for healing, poisons and stat changes done by using the INT and FLOAT stat modifiers, it can completely destroy a character if custom effects are being run that also do not get reset on the player entering the server.

An example: When we began implementing the temperature buffs for the highlands map expansion, we implemented a buff that affected the penis size of male players while in cold weather. Back then, we didn't have a stat for this and so in the buff, we accessed the character creation component of the character and modified the stat there.

This had the effect that when a player entered a cold weather zone, the penis would shrink. The player then disconnected and reconnected - with a penis size that was now smaller. And immediately got affected by the cold weather zone, thus shrinking the penis further. This would repeat itself until the size of the penis reached negative numbers and while it might sound funny, eventually destroyed the possibility to play with the character.

In other words - be very careful about what a buff does. Test your buffs by applying them and logging out while they are still running, then start up again and see if the effects are retained.

Make use of the parent-call functions

I cannot stress enough the importance of calling the parent function for buffs. This is the most common mistake, hands down, that (while not terribly dangerous) makes buff not do what they are supposed to.