Table of Contents

About this primer

This primer will walk you through the Modcontroller, how to set it up and how to work with it. It will not dive into any of the tables or what they do - there are other guides for those topics. If you are already familiar with the mod-controller, this guide is likely not going to have many surprises, but if you are new to modding Conan Exiles, this primer is for you.

What is the Mod-Controller?

The mod-controller is a central hub for your mod where you can attach your new snazzy data-tables to core game data-tables (required for making new items, recipes, feats, or any other data table-work). It also allows for attaching components to blueprints, something which is very useful if you want to add new functionality to certain parts of the game.

In essence - it makes certain that your mod gets loaded correctly.

What does the Mod-Controller do?

When Conan Exiles loads the game with mods installed, it automatically checks for the presence of Mod-controllers; it then proceeds to go through the mod-controllers and merge all tables for all mods as well as attach any blueprint-components set up in the controller.

Quick recap on how to create a mod

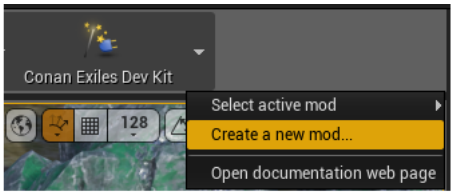
Before we get into the mod-controller, it may be useful to know how to actually create a new mod using the Conan Exiles mod-kit.

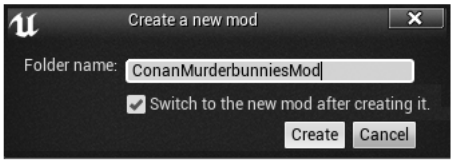
The first thing you will need to do is to click the Down-arrow next to the Conan Exiles Dev Kit icon in the top right part of the screen.

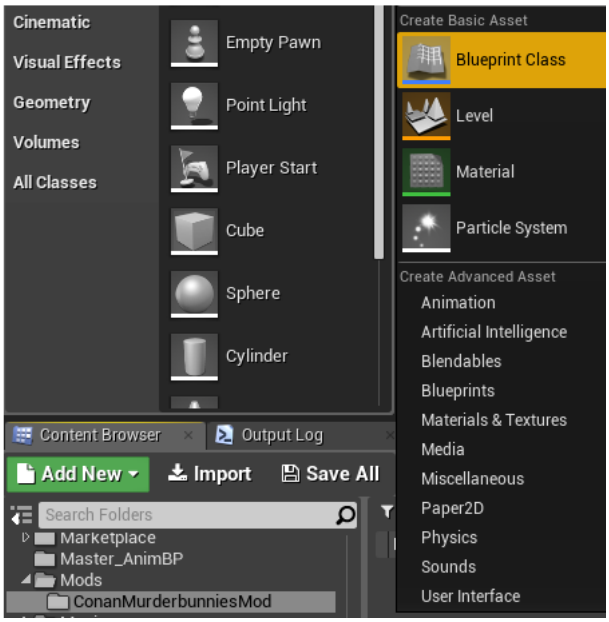
This will bring down a menu from where you can select "Create new mod".

Clicking this will give you a input-box requesting a name for your mod.

Once you have selected a name and clicked "Create" here, the Conan Exiles Mod-kit will re-start itself and start up with your mod being the one loaded, so that you can work on it.







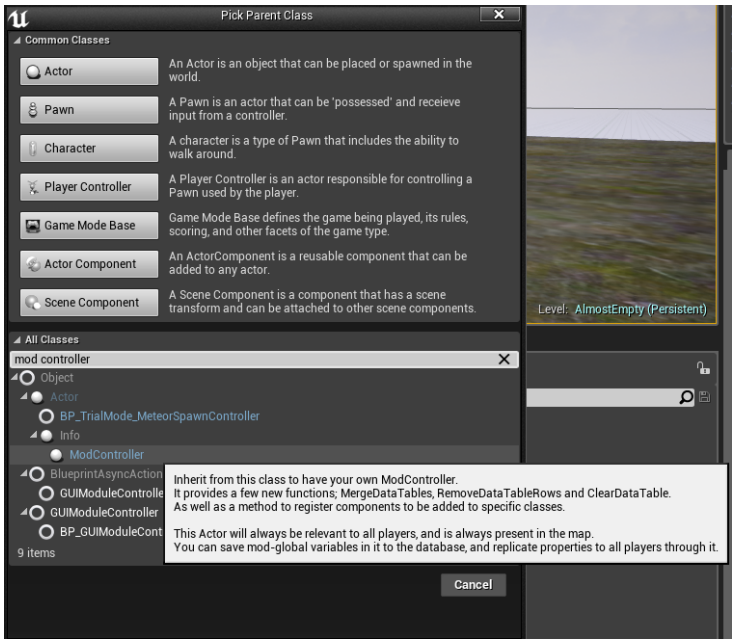
Adding the Mod-Controller

Assuming you have now created a mod, you will need to find the folder for that mod, right-click somewhere inside the asset browser, and select "New Blueprint Class".

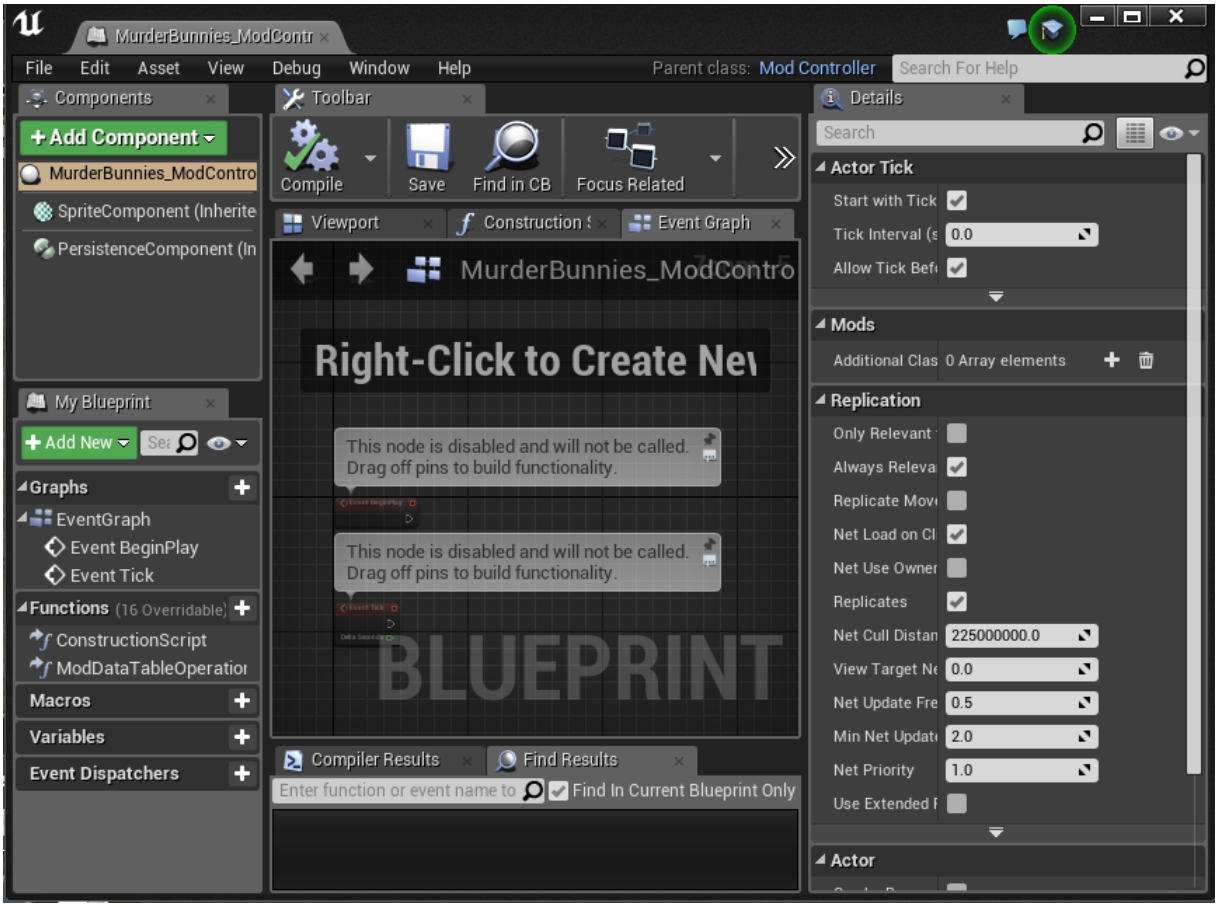
You will get a selector that allows you to pick between a number of common classes and you will also get a search field.

If you enter "Modcontroller" into this search field, you will find the correct entry.

You can name your Mod-Controller anything you like. In our case, we are naming the mod-controller "MurderBunnies_ModController".



When you open up your mod-controller, it will look something like the image below.

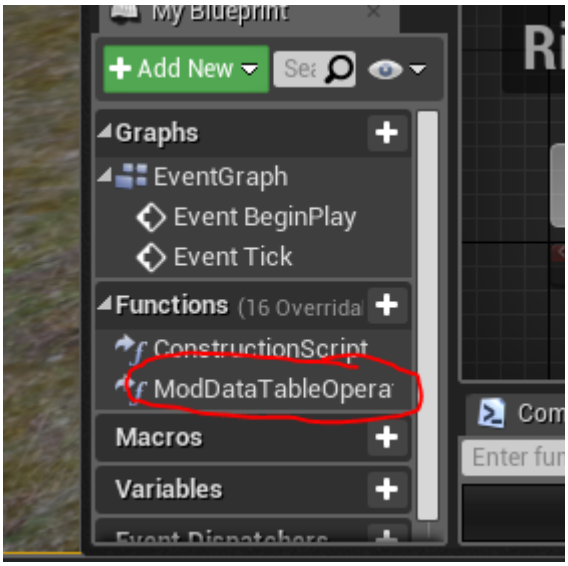


Adding Data-Tables

If you want to add new recipes, items, feats or work with any other table, you will need to make a new table of the same type. This new data-table will then (by the power of the mod-controller) be merged into the basegame table and give access to all of your entries.

For a guide on the different tables, please refer to the "Struct Reference document" guide in this .zip file. For the examples below, we will be adding the three most common tables: The ItemTable, RecipesTable and FeatTable.

After adding the three core tables, the contents of our mod-folder looks like this. We've added a FeatTable, a RecipesTable and an ItemTable. Now we need to get the mod-controller to actually merge those together when we build our mod.

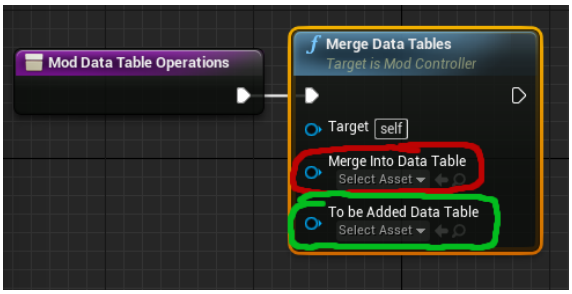


By double-clicking on the "ModDataTableOperations" function in the Functions list of the Mod-controller, you will bring up a new tab in the blueprint. In this tab you will see a purple node denoting the start of this function.

If you left-mousebutton and drag out from that node, you will get a list of possible calls you can make using this function; the one that we are looking for is called

"**Merge Data Tables**"

When you click this, a new blue node will appear, where there are two drop-down selections - one for the Table to be merged INTO (the base game table, marked in red in the image on the right) and the other one for the table to BE merged (your mod table, marked in green in the same image).

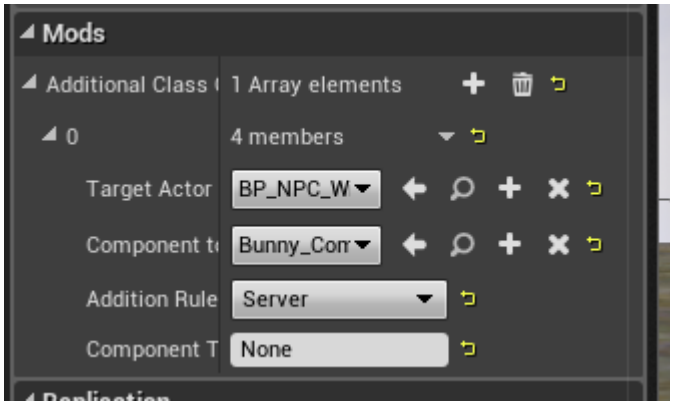


You will need to set up each table you want to merge this way.

Adding Components to blueprints

In order to access certain functions of many blueprints, it's better to attach a component to the blueprint rather than modifying the original asset. If anyone else edits the core game asset, the mod load order will simply apply the last-one-in principle, and your functionality may/will be lost. For this reason, the mod-controller can add components to blueprints.

In order to attach a component, you first need to create the component. As with creating the Mod-controller, you will need to right-click in the Content Browser. This time, select "**Blueprint Class**" and in the Parent Class picker, select "Actor Component".



In our example, we'll create one called "**Bunny_Component**" and attach it to the actor "**BP_NPC_Wildlife_Rabbit**" blueprint.

It's worth noting that the Addition rule is very important. For functionality that runs on the server (default setting) things are straightforward enough. For information that needs to be communicated from clients to servers and vice-versa, use the Server and Client Copies, and finally for functions that only need to happen on the client, use "Client". Knowing what to use here is a matter of experience and knowing the systems and a full guide on this would be out of scope for this primer (sorry).

You have now set up a basic mod-controller complete with tables and a component.

For working with components there are more extensive guides than this one - here is the Unreal Engine official documentation for Actor Components:

https://docs.unrealengine.com/en-US/Programming/UnrealArchitecture/Actors/Components