ARCHIVEXL ITEM ADDITION TUTORIAL

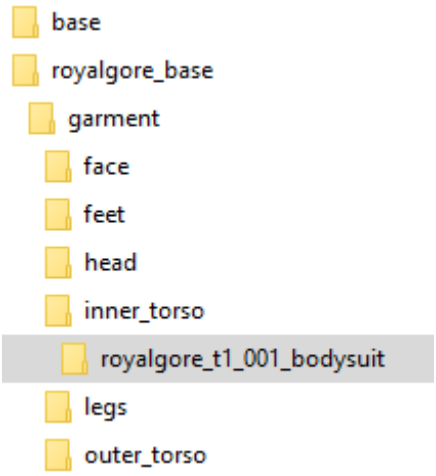
CREDITS

- Royalgore - I wrote this!
- Psiberx - the creation of ArchiveXL AND TweakXL.
- Johnson - For the HUGE amount of patience and hilarity that went with learning how to do this in the first place.
- Nim - For being a Chad ™ and all the previous work with CSVmerge that led to this version being possible. Certain bits of information I learned directly from Nim, some I learned from the person below, it's hard to distinguish it now, but they're both very important and very cool people.
- Halvkyrie - For moral support and for all the information I got from reading your messages in the discord even when you weren't talking to me.
- 123321 - For giving me that little bit of information that instantly spawned five serotonins in my brain and cured my depression.

Disclaimer: This tutorial assumes that you are already aware of and capable of doing basic modding procedures not limited to mlsetup editing, mesh editing, mesh replacements, and hex editing. There won't be super intense things needed, but the assumption that these are within your capabilities is important. You should already know what the majority of the resources are that are used for CBP2077 modding, but here is a list of what is mandatory / useful.

- Wolvenkit Stable build OR Wolvenkit Nightly build
    - Buffer order was changed between 8.5.0 and 8.6.0 / 8.6.0 Nightly. As a precaution, use the same wkit.cli / gui when editing a singular file AT ALL TIMES, or risk submesh disorganization or just flat out breaking the item.
- Mlsetupbuilder.
- 010 Editor.
- ArchiveXL + RED4ext + tweakXL triad.
- TBA. Honestly if you need to know what other stuff you need, you probably need to spend more time modding before trying this out. :>
- **The project made with this tutorial** ; use it as a template or use it as a source to learn from.

PRE GAME YOUR MODDING WITH MORE MODDING.



Custom path the items you're adding / all the files relevant to the item you're adding. This tutorial will only be adding an existing NPC item into the game as something V can wear. I won't be covering editing these meshes, as that is a whole different ball game and something you should learn how to do long before you start making additions.

I will be adding the bodysuit to the game as a separate item for Fem V. To do that, I acquire the mesh necessary, and move it to a custom folder of my choice. I usually follow a similar folder system that CDPR used.

My .mesh file is in the highlighted folder. If I want to do custom mlsetups, I would make a "textures" folder inside of it and keep xbm / mlsetups / mlmasks there. If you're editing meshes, changing textures, etc– you do this during this step, and should do an item replacement to ensure it works properly.

I intend on making this compatible with non-vanilla Fem V bodies, so custom pathing is absolutely necessary unless I want to break the mesh on NPCs.

THE ENTITY - APP - ENTITY - MESH CHAIN

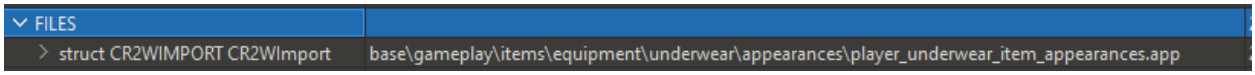
The project, if downloaded, already includes this chain simplified and tailored for this addition. However. I labeled them A, B, and C because they connect in that order. To make this chain, I did the following:

1. royalgore_base\item_addition\royalgore_t1_001_bodysuit\
    a. This is where I am putting the .ent and .app files.
2. base\gameplay\items\equipment\underwear\player_underwear_item.ent
    a. I moved player_underwear_item.ent and renamed it to a_rootentity.
3. base\gameplay\items\equipment\underwear\appearances\player_underwear_item_appearances.app
    a. I moved player_underwear_item_appearances.app and renamed it to b_appearanceapp.
4. base\characters\garment\player_equipment\torso\t1_057_pwa_tank__bra.ent
    a. I moved t1_057_pwa_tank__bra.ent and renamed it to c_meshentity.

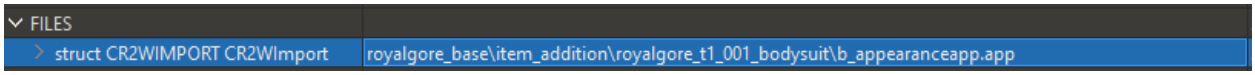
YOU DO NOT HAVE TO MOVE THESE FILES, BUT YOU DO HAVE TO RENAME THEM AND NOTE THE NAMES DOWN. In my experience, most people will add their modder name to it, then something else that will allow them to identify what it is easily. A lot of modders also keep them similarly named / identically named but in different folders ; I find this confusing, so I don't do that, and I put my chains in a place where I can find it easier.

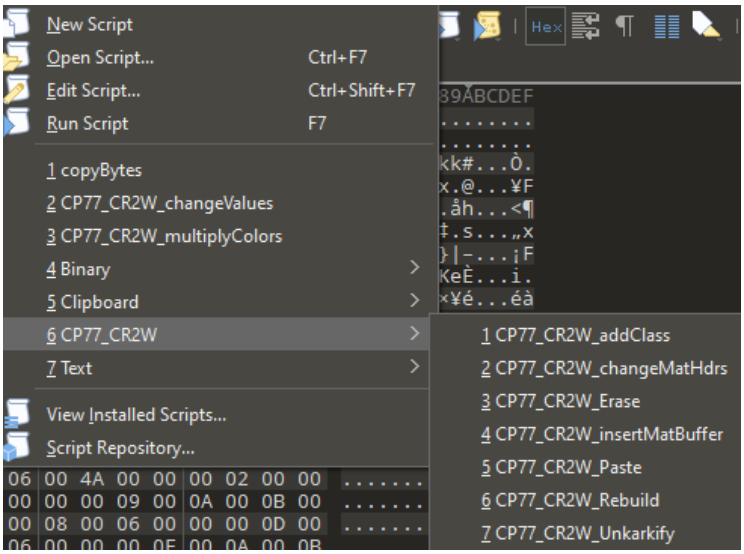
I also don't do most if not any of my edits through WKIT GUI, so I cannot cover how to do that specifically. 010 Editor allows more flexibility and total control over the file which is why I prefer it, but you will more than likely be able to translate changes here into the GUI.

With all files collected, I will start by editing a_rootentity first. Expand the "FILES" section and change the path to match b_appearanceapp.



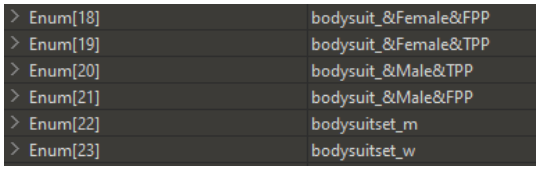
TO





From there, open CNAMES, run the "Paste" script to add entries here. Make sure you have Cnames selected so you don't paste anything in the wrong area. I added the following lines:

bodysuit_&Female&FPP
bodysuit_&Female&TPP
bodysuit_&Male&TPP
bodysuit_&Male&FPP
bodysuitset_m
bodysuitset_w



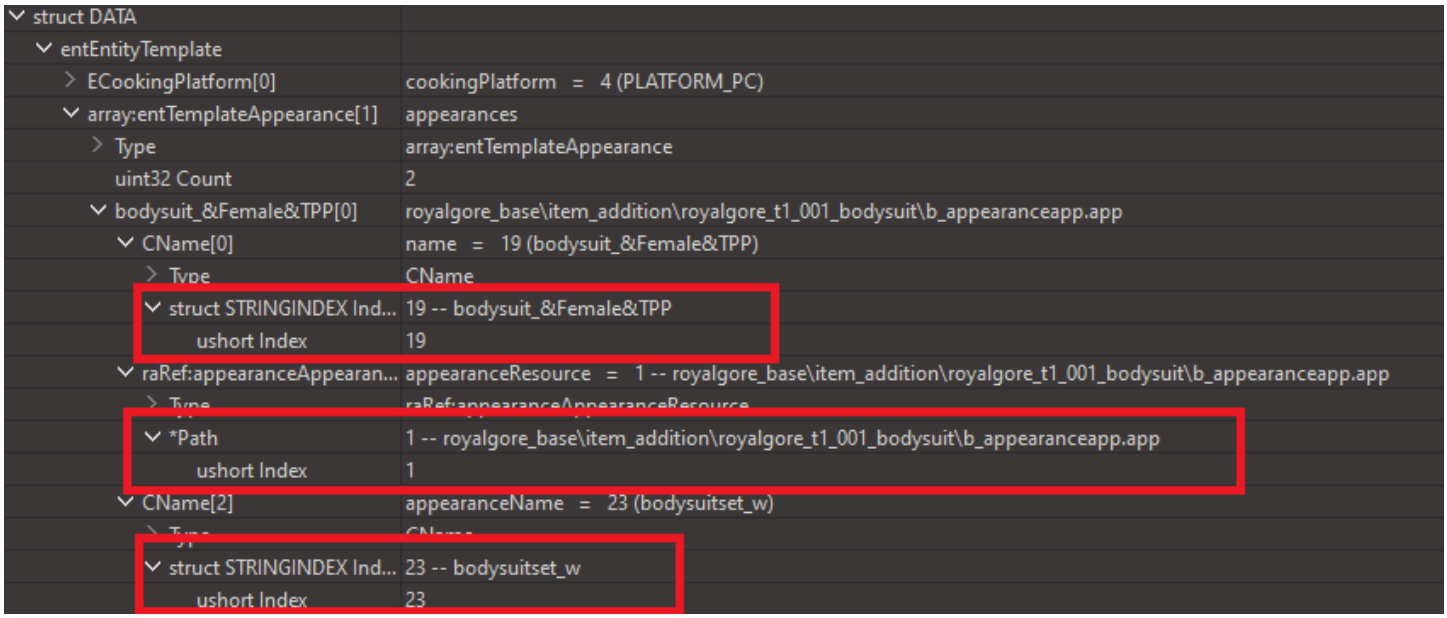
Disclaimer: You can skip this step technically and use pre-existing cnames from ents, but this one didn't have the ones I needed, and I prefer to do it this way since I will absolutely remember what I used. This is down to personal preference so long as it meets the criteria to the right!

PERSPECTIVE SUFFIXES USED PER SLOT

Outer : &FPP / &TPP
Inner : null / &TPP / &TPP&Full / &TPP&Part
Outfit : &FPP / &TPP
Legs : null / null
Head : &FPP / &TPP / &TPP&Bald / &TPP&Short &TPP&Long / &TPP&Dreads / &TPP&Buzz
Feet : null / null
Face : &FPP / &TPP
null : &Female / &Male

These suffixes are referenced when the item is displayed, and need to be included in the cname. For a torso item, specifying the body type (masc or fem) and First person perspective / third person perspective is necessary. This will make it so you can load a different appearance in first person and a different one in third ; CDPR uses this to roll up the sleeves of all shirts, for example.

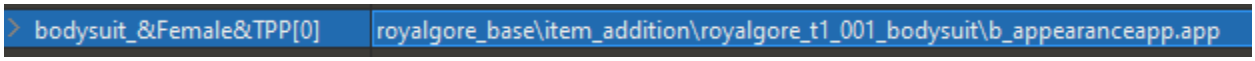
After adding those cnames, go further below and expand StructData and entEntityTemplate. Inside there, array:TemplateAppearance contains all the chain information that connects a_rootentity.ent to b_appearanceapp.app. For mine, I only have one linked file in the FILE section, so it will already be correct here, but the hierarchy is pictured below :



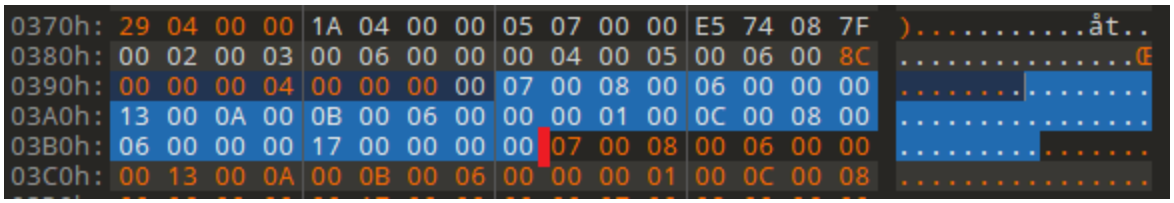
Change CName values to the respective numbers or copy and paste the values you put in before. Using the cname reference numbers is the easier way, so keep that stuff noted down. I recommend making the cnames a little different so you can tell what's actually changing here. With the Cnames I had already, I changed the ushortIndex values in the red boxes and solidified my changes. Save the file at this point, then open b_appearanceapp.app

OPTIONAL / WHAT TO DO IF YOU DON'T HAVE ENOUGH LINES HERE

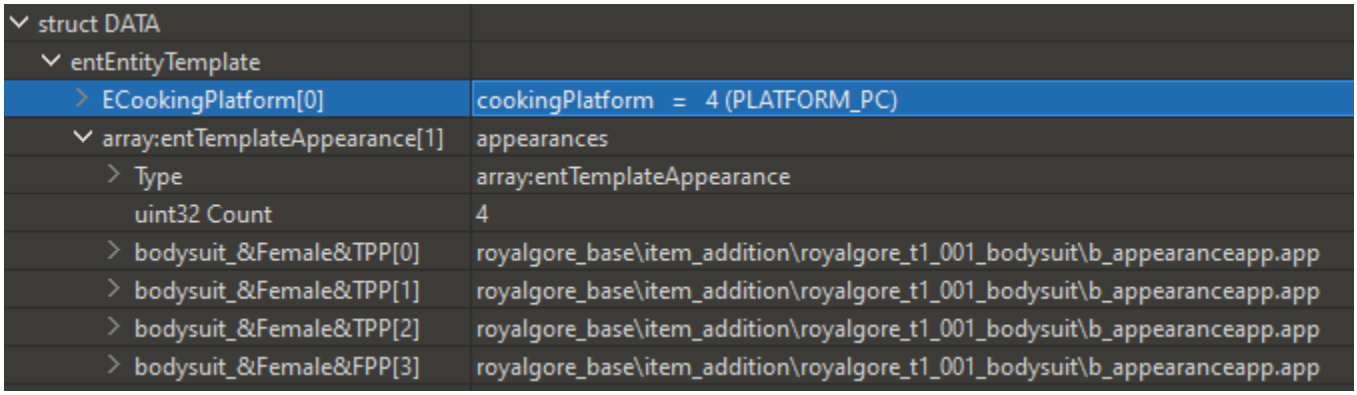
If you don't have enough lines here, click on the line without it expanded.



In the raw data, it will highlight a portion. Right click and copy it, then put your cursor point at the last number in the line. Make sure you if you type anything, it will be AFTER the last highlighted number.



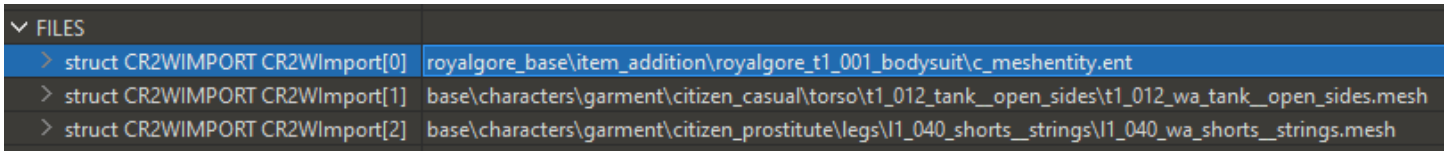
Use the script to paste, and you will create more lines here. If you did it correctly, you will have duplicated the section and can now edit it freely as you wish. Mine only had two, but it now has four, seen here.



struct DATA	
entEntityTemplate	
ECookingPlatform[0]	cookingPlatform = 4 (PLATFORM_PC)
array:entTemplateAppearance[1]	appearances
Type	array:entTemplateAppearance
uint32 Count	4
bodysuit_&Female&TPP[0]	royalgore_base\item_addition\royalgore_t1_001_bodysuit\b_appearanceapp.app
bodysuit_&Female&TPP[1]	royalgore_base\item_addition\royalgore_t1_001_bodysuit\b_appearanceapp.app
bodysuit_&Female&TPP[2]	royalgore_base\item_addition\royalgore_t1_001_bodysuit\b_appearanceapp.app
bodysuit_&Female&FPP[3]	royalgore_base\item_addition\royalgore_t1_001_bodysuit\b_appearanceapp.app

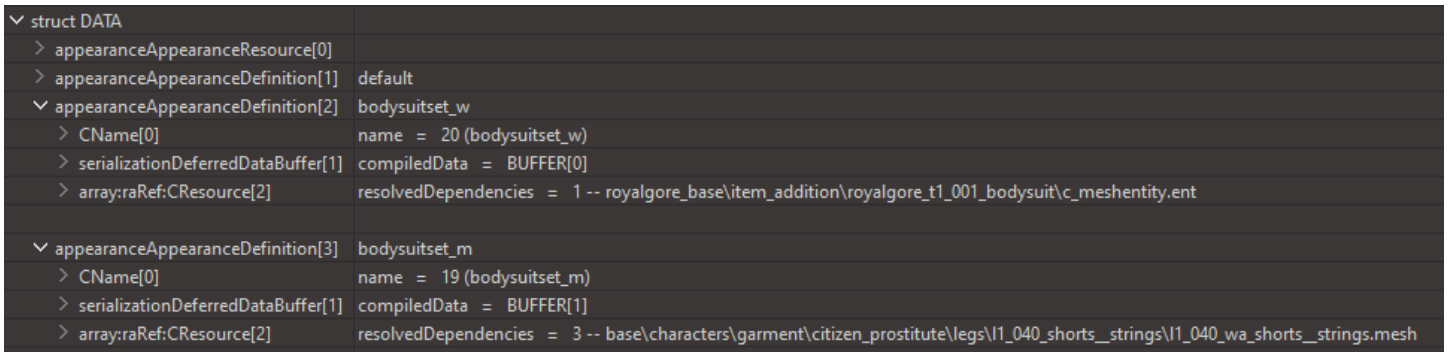
I don't HAVE to do this, but I did anyway ; this won't be compatible for masc V, but I will still change two of those lines to mirror masc V for anyone who wants to use the template. If you are only doing additions for one body type, you can skip this part. If the item is exactly the same for both body types, you can reference the same lines for all of them but have different section names for the game to refer to.

After saving a_rootentity.ent , open b_appearanceapp.app. Again, in the files section, you will need to change at least one of these lines to reference c_meshentity.ent.



FILES	
struct CR2WIMPORT CR2WImport[0]	royalgore_base\item_addition\royalgore_t1_001_bodysuit\c_meshentity.ent
struct CR2WIMPORT CR2WImport[1]	base\characters\garment\citizen_casual\torso\t1_012_tank__open_sides\t1_012_wa_tank__open_sides.mesh
struct CR2WIMPORT CR2WImport[2]	base\characters\garment\citizen_prostitute\legs\l1_040_shorts__strings\l1_040_wa_shorts__strings.mesh

In CNAMES, I will be adding the following lines again: bodysuitset_m and bodysuitset_w, noting down the numbers that refer to it. Like before, I expand struct DATA and change lines similarly to how I did for a_rootentity.ent.

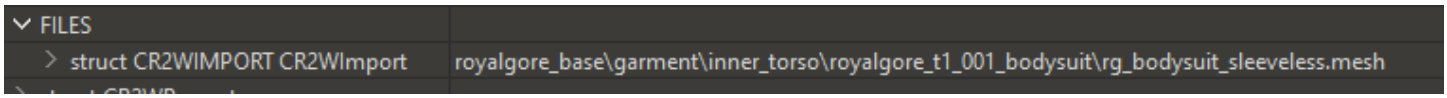


struct DATA	
appearanceAppearanceResource[0]	
appearanceAppearanceDefinition[1]	default
appearanceAppearanceDefinition[2]	bodysuitset_w
CName[0]	name = 20 (bodysuitset_w)
serializationDeferredDataBuffer[1]	compiledData = BUFFER[0]
array:raRef:CResource[2]	resolvedDependencies = 1 -- royalgore_base\item_addition\royalgore_t1_001_bodysuit\c_meshentity.ent
appearanceAppearanceDefinition[3]	bodysuitset_m
CName[0]	name = 19 (bodysuitset_m)
serializationDeferredDataBuffer[1]	compiledData = BUFFER[1]
array:raRef:CResource[2]	resolvedDependencies = 3 -- base\characters\garment\citizen_prostitute\legs\l1_040_shorts__strings\l1_040_wa_shorts__strings.mesh

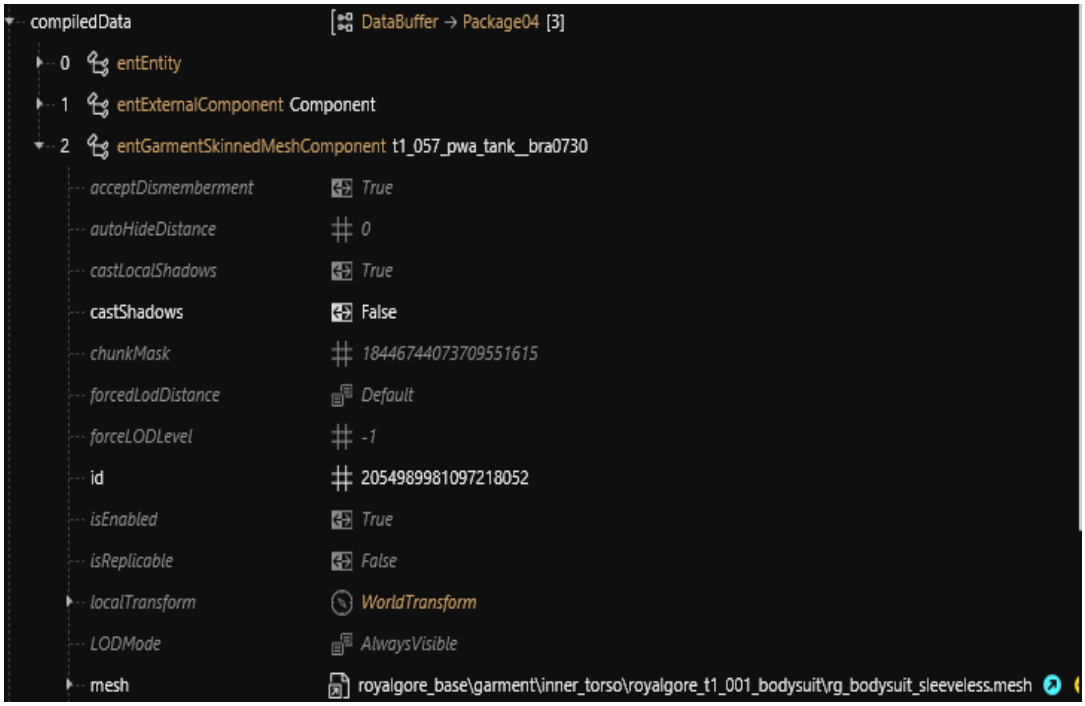
I change the cname to match the lines from the file prior. This connects the two and allows the first one to point at the data specified by it. You CAN link directly to a mesh, but I prefer to link to an entity first, as I can link multiple meshes in one entity instead. Since I don't have a masc V version of this bodysuit, I leave it as .mesh.

If masc V equips the item it will load thhe mesh specified, but it will not be correct because it's a Fem V fitted item. Therefore, it's not compatible with masc V, but technically still works.

Save and move on to c_meshentity.ent. You will practically repeat the same steps as before, except this time, this ent directs to the .mesh itself.



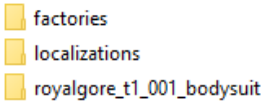
The easiest way to do the final edit on this is to export it into json with the commands in the next section, then change the filepath in compiledData as well. Import that, and it'll be properly set up. If you don't want to do that either, you can open WKIT GUI and edit it here :



The chain is done. There will be additional info later on about loading different appearances of that mesh, but for now we will move on and make sure the rest of the files are set up properly + showing up in game the way it needs to be.

CSV, XL, Localization, AND YAML

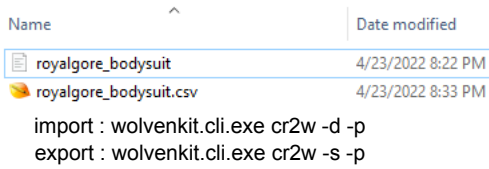
We will be doing the CSV first, but it's also nice to collect all the files you'll need. In the game directory, find and make a copy of the files listed below. I move mine as well, but like the ents and app files, you don't have to, so long as you remember where it is and what it is named.



base\gameplay\factories\items\clothing.csv
base\localization\en-us\onscreens\onscreens.json

I moved these files into separate folders for organization. I plan on adding more to these folders with future mods, so having a place where I can find them all will come in handy for me.

Open wolvenkit.cli through the cmd. I am using version 1.6.5. It might be the nightly version, it might not be, I can't recall, but the important thing is to use the same version for both importing and exporting with these files because the way the json was converted has changed and is no longer compatible between versions. Yours will look a little different than mine for this reason if you are using 1.5.2. The commands are to the right.



I will edit clothing.csv first, which I renamed to royalgore_bodysuit.csv.

Open the json file that is made by doing this. Any text editor works, but I use visual studio code, since it will point out any errors in my json if I make them.

From there, delete all the entries under "Compiled Data" and before "headers."

```
D:\cyberpunk_modding\006_wolvenkit.cli\1.6.5>
D:\cyberpunk_modding\006_wolvenkit.cli\1.6.5>wolvenkit.cli.exe cr2w -d -p D:\cyberpunk_modding\001_current_projects\royalgore_leotard\source\archive\royalgore_base\item_addition\factories\royalgore_bodysuit.csv.json
[ 0: Information ] - Found 1 files to process.
[ 0: Success     ] - Converted D:\cyberpunk_modding\001_current_projects\royalgore_leotard\source\archive\royalgore_base\item_addition\factories\royalgore_bodysuit.csv.json to CR2W
[ 0: Information ] - Elapsed time: 3909ms.
```

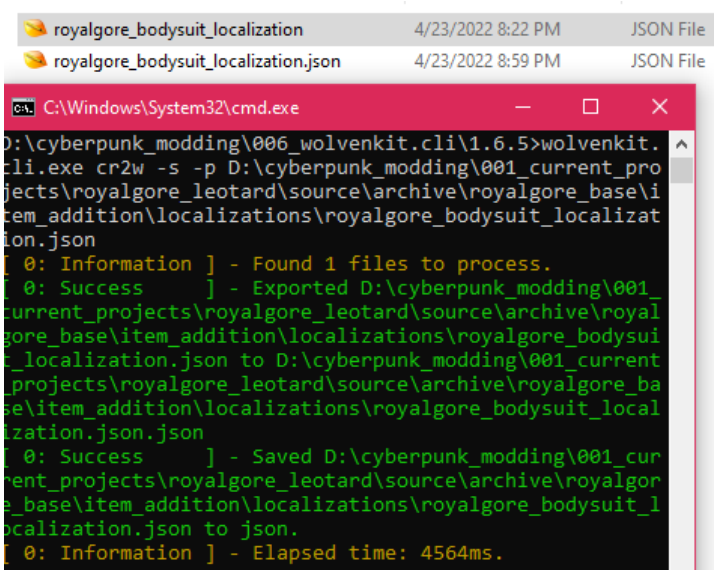
```
"Data": {
  "Version": 195,
  "BuildVersion": 0,
  "RootChunk": {
    "Type": "C2dArray",
    "Properties": {
      "compiledData": [
        [
          "clothing_root",
          "user\\christopher_schulte\\clothing_root.ent",
          "true"
        ],


        [
          "royalgore_bodysuit",
          "royalgore_base\\item_addition\\royalgore_t1_001_bodysuit\\a_rootentity.ent",
          "true"
        ]

      ],
      "headers": [
        "name",
        "path",
        "preload"
      ]
    }
  },
  "EmbeddedFiles": []
}
```

This is an example of the entries. I put a red line next to one I didn't edit, and a pink line next to my entry. I will delete the red entry as it serves no purpose, but it will show how to edit it properly.

Ensure the brackets are properly closed off and there are no errors in the json.

Use the import command to import the changes to the csv.

```
[
        "clothing_root",
        "user\\christopher_schulte\\clothing_root.ent",
        "true"
      ],
```



Now, I will edit the onscreens.json file that I renamed to royalgore_bodysuit_localizations.json. Like the csv, you will need to export it into a legible json to edit it. Don't change the name of it , it needs to be .json.json.

Open it with your text editor and make changes similar to the last. This file defines the item name and description. I am not particularly good at describing things so I did mine very lazily. This file has a LOT of text, so I recommend wiping it out with the following directions:

Copy the lines above "Data". Delete ALL text in the file after, then past the header back in. Do a couple of lines and close the bracket.

Left before ; right after.

The template to the right can be pasted and edited from there. (Double check to make sure the version info from the template matches the version info on the json you started. I'm not sure if this will cause issues, but better to check anyway.)

The template has two entries. the red text is the reference ID for it, while the orange text is the information that will appear on the item description or title. How you label these is up to you, but I tend to use _desc and _title so I can recognize it better. "MaleVariant" and "femaleVariant" dictate the information that is shown, and "null" means it will not have anything there if your V matches that body type.

If you want to make variations of the item, you can add more entries and point at them in the next steps. If you want them to be identical, then you can refer to it this way too.

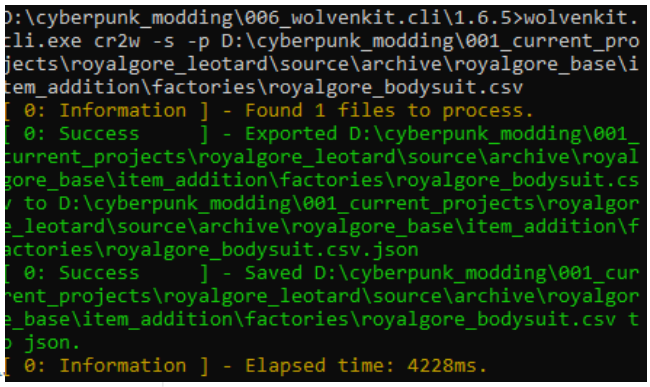
VSC double checks my lines for me, and these are all correct. From there, I will save it, and import it back into the original json.

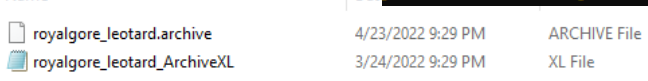
The original localization file was 7,400kb. It will now be 1 or 2kb, typically. That way, you can tell for sure if it imported.


```
"Data": {
  "Version": 195,
  "BuildVersion": 0,
  "RootChunk": {
   "Type": "JsonResource",
   "Properties": {
    "cookingPlatform": "PLATFORM_PC",
    "root": {
     "HandleId": "0",
     "Data": {
      "Type": "localizationPersistenceOnScreenEntries",
      "Properties": {
       "entries": [
        {
         "Type": "localizationPersistenceOnScreenEntry",
         "Properties": {
          "femaleVariant": "RG - Bodysuit",
          "maleVariant": null,
          "primaryKey": 0,
          "secondaryKey": "royalgore_bodysuit_name"
         }
        },
        {
         "Type": "localizationPersistenceOnScreenEntry",
         "Properties": {
          "femaleVariant": "Sorry, I am not descript. - RG",
          "maleVariant": null,
          "primaryKey": 0,
          "secondaryKey": "royalgore_bodysuit_desc"
         }
        }
       ]
      }
     }
    }
   }
  },
  "EmbeddedFiles": []
```


The next thing to do is the XL file that will be in the directory with your .archive file once it's packed. Everything that goes in the .archive is complete, so you can pack it up and put it in a group of folders that'll allow an easy installation. Since I didn't null any garmentsupport, I can use the GUI to pack this up, and the directory in the "packed" folder will already be correct.

Right click in the folder with your packed archive and create a new txt file. Change the extension to XL, and name it similarly to the archive file itself for organization's sake.





Open the XL file. Paste the text on the right into it, and correct the paths to direct to the CSV and Localization files you edited earlier. Save this file, then you're done with it.

```
factories:
  - base\gameplay\factories\items\clothing.csv
localization:
  onscreens:
    en-us: base\localization\en-us\onscreens\onscreens.json
```

```
factories:
  - royalgore_base\item_addition\factories\royalgore_bodysuit.csv
localization:
  onscreens:
    en-us: royalgore_base\item_addition\localizations\royalgore_bodysuit_localization.json
```

The final file you will need to do is the YAML file. This will go here :

r6\tweaks\royalgore_yaml\royalgore_bodysuit.yaml

You can use my yaml file provided, or use this text to the right as a template.
Blue text is the item ID for spawning it in.
EntityName is the name you gave the .ent in your csv file.
appearanceName is the name you gave in a_rootentity.
Display name is the title ID you made in localization.
LocalizedDescription is the description you made in localization.
Icon's parts are not covered yet, but are done similar to the other files, just with an .inkatlas file and xbms for the icons. (will cover soon.)

```
Items.royalgore_bodysuit_001:
  $base: Items.GenericInnerChestClothing
  entityName: royalgore_bodysuit
  appearanceName: bodysuit_
  displayName: royalgore_bodysuit_name
  localizedDescription: royalgore_bodysuit_desc
  icon:
    atlasResourcePath:
royalgore_base\item_addition\iconatlas\royalgore_bodysuit.inkatlas
    atlasPartName: icon_id_001
```

I will add more information on lines that can be added here soon.

TROUBLESHOOTING

If you go in and encounter problems, they can be solved a number of ways, but lets start with some of the most common:

- Item doesn't spawn - YAML error. Double check your YAML file.
- Item spawns, but no description / title - localization error. Double check localization is filled out properly AND check the .xl file to make sure it is pathed correctly.
- Item spawns and is equippable, but shows no meshes - a more complicated issue. This one will have a few different suggestions below:

Suggestion A

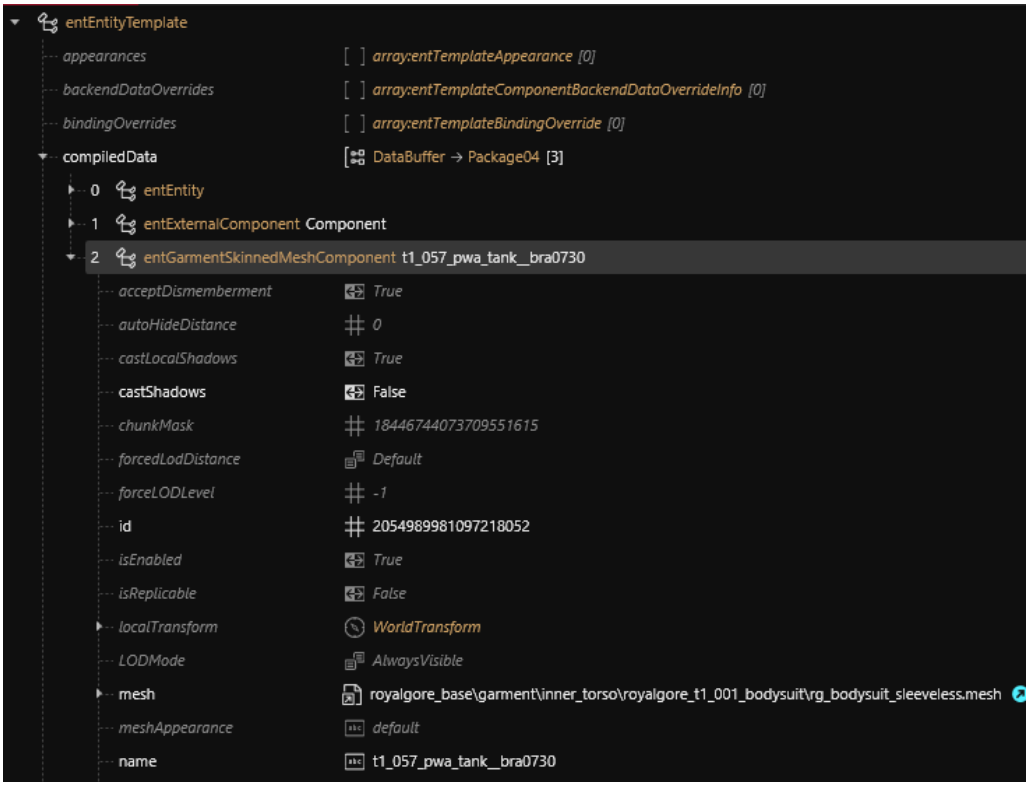
Use the same mesh to replace an ingame equippable. if it loads, then your mesh is not the problem.

Suggestion B

Double check YAML name references, and make sure the CSV is also correct. After that, ensure your entity - app - entity - mesh chain is properly pathed.

Suggestion C

Open up the last ent in the chain in wkit GUI. Expand compiledData, then ensure the mesh path is pointing to the correct mesh. The "name" doesn't matter, but you should change that too. Also, check LOD mode in the app and make sure its set to "AlwaysVisible."



Future plans

Expansion on suffixes & general data that may prove useful.
Inkatlas and icon creation / guidelines.
Add appearance versions to the game as well.

Walkthrough through the wkit GUI ; it's practically the same thing, just more user friendly and harder to mess up. If you can do it through 010 editor, you can definitely do it through the GUI!