Repository Clothes using the Repository Wizard (duh)

Disclaimer: This tutorial focuses on teaching how to use the Repository Wizard (RW) to repository clothes, as well as some tips on how to use it to our advantage for packaging them.

Note: Sections are hidden by default. Click the little arrow icon next to their titles (left side) to show/hide a section. I recommend you read through the entire "**Before you start**" section if it's your first time using the RW or one of whoward's utilities.

Before you start

What is the "repository" technique?

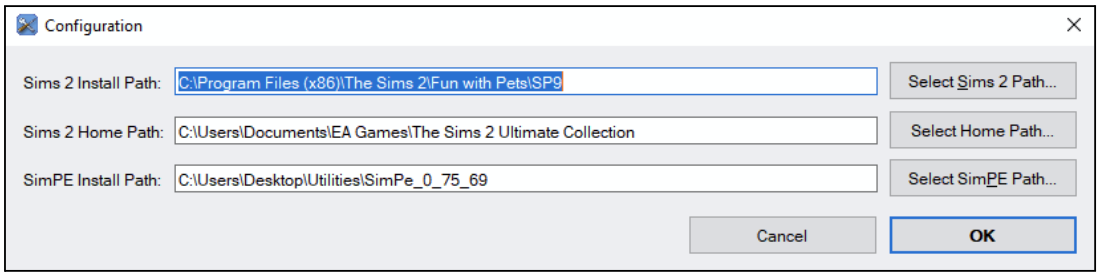
"Repositorying" is the process of making different files share the same resources, mainly texture related ones. It is also known as "slaving" or the "master/slave" technique. Doing this results in smaller file sizes since there's less resources to load. Maxis itself used this technique all the time for that reason.

Unfortunately, the creator tools available before the Repository Wizard didn't make this process very easy to do. References to shared resources had to be edited manually, which meant going through all the packages that use those shared resources one-by-one. Very time consuming!

The Repository Wizard automates the whole process, optimizing it further, and even lets you de-repository those same items, so that you can provide a "Standalone" alternative within seconds.

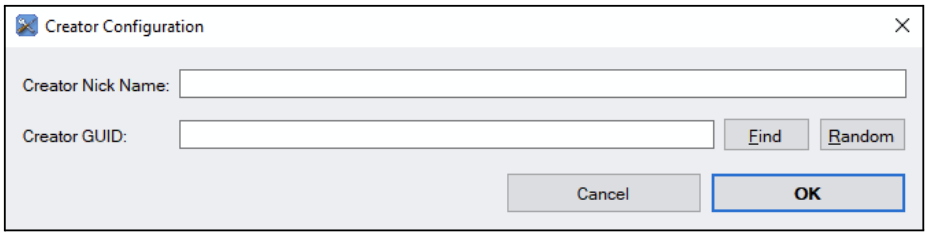
Setting up

Open the RW, go to **File > Configuration** …



… and select the path to all three directories, as requested.

Then, go to **File > Creator Details** …



… and type in the required information.

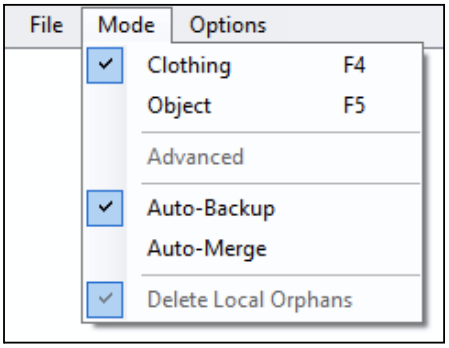
The **Creator GUID** is a randomly assigned value that Body Shop generates for you each time you install the game. You can either: copy-paste it, use the “Find” button to retrieve it automatically from the Projects folder (requires at least 1 existing project), or generate a new one with the “Random” button.

The RW will use this as the “Creator” value when generating new package files.

Since we are focusing on Clothes in this tutorial, we also need to select the correct mode.

Navigate to the **Mode menu, and select “Clothes”**.

By default the program also enables the automatic merging of files, which might not be what you want if you plan on sharing those files, so in the same **Mode menu, uncheck “Auto-Merge”**.

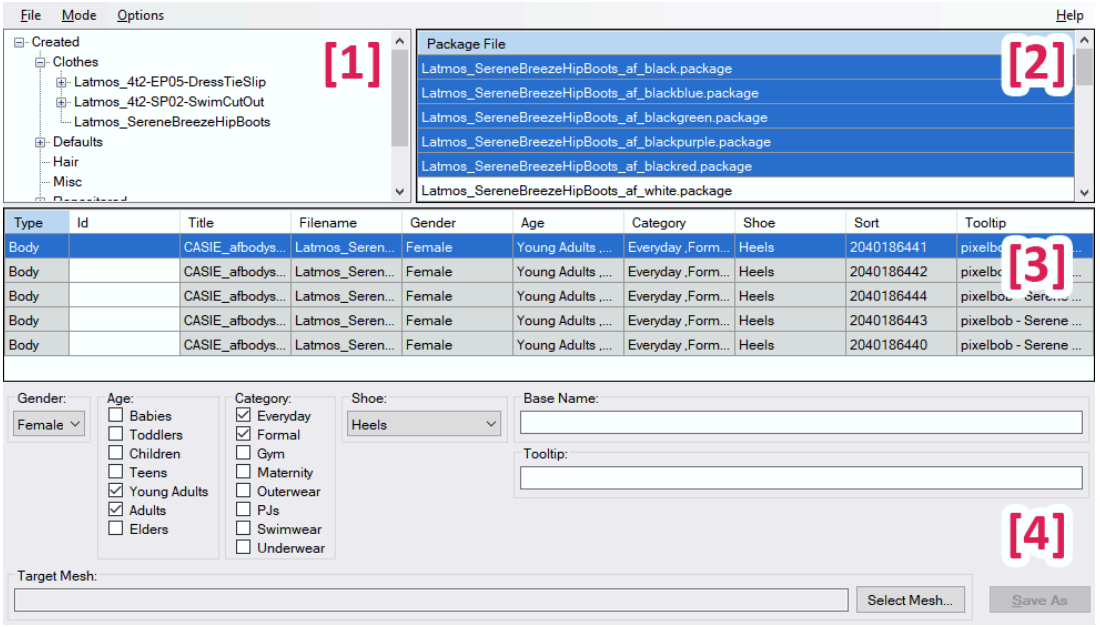


Keep in mind that these will remain the same in-between sessions, so you might want to switch them from time to time depending on your needs.

Rundown

Note: In this tutorial, the word “item” refers to an entry in the CAS catalog, while “file” refers to the actual package file. Therefore, a “recolor ***file***” is the package that contains the recolor ***item***.

Usually there is only 1 item per package (that’s how Body Shop saves them). But even if you were to load a merged package in the RW, it would still display all the items it contains just as well.



The structure of your loaded folder appears on the left [1], while the package files the active folder contains appear on the right [2]. If the loaded folder has subfolders, you can make them active by simply selecting them on the left [1], which will update the list on the right [2].

Once some files have been selected, the items they contain will appear on the table below [3].

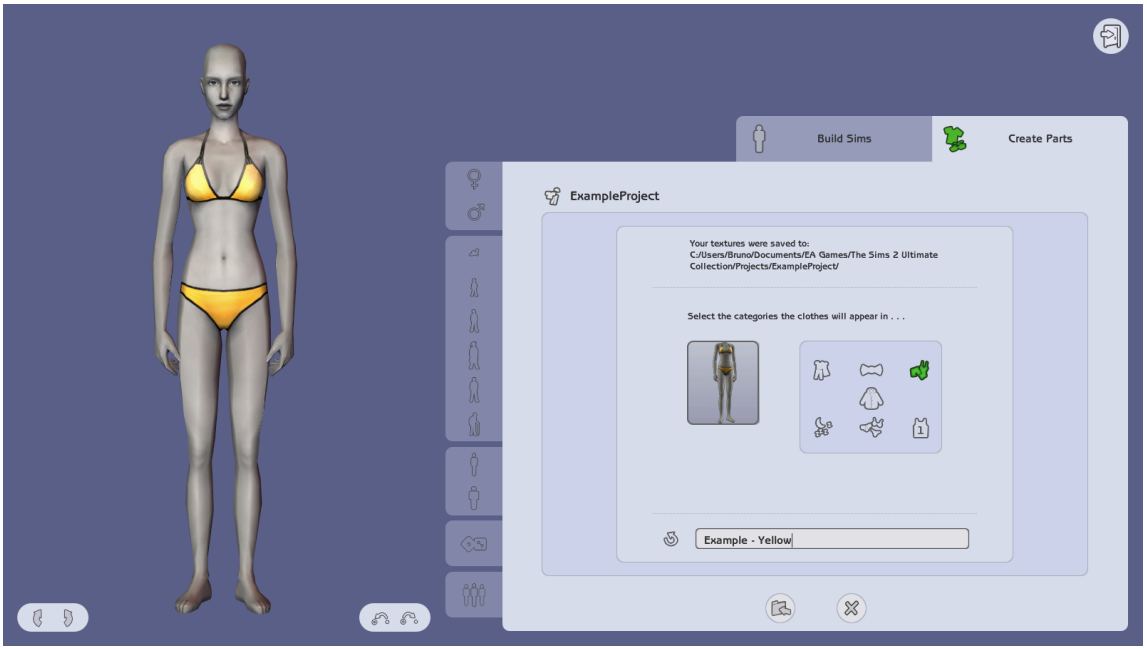
The last section [4] will display the gender, age, category and shoe flags from the selected items (if items have different flags, the RW will display it differently to reflect that). Those will be used for the repo'd items, so change them if needed! Base Name and Tooltip will remain empty until you set them.

Basic usage

How to repository

This part assumes that you know how to use Body Shop to make recolors and that you use SimPE to import the textures (and not Body Shop).

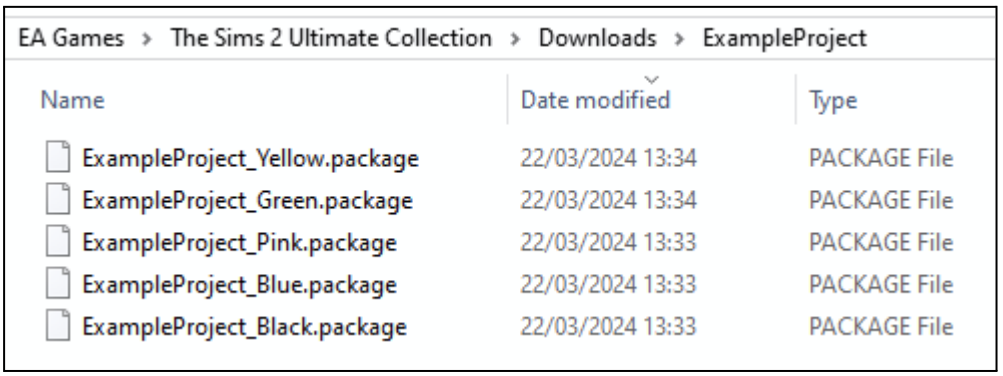
1. Open Body Shop and proceed to make as many recolors that you need.



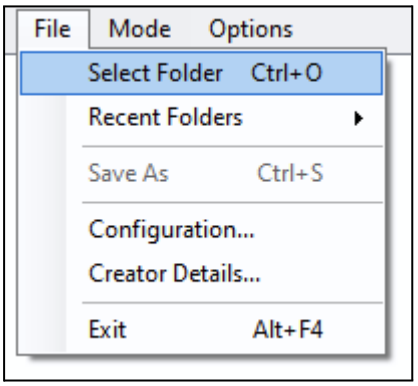
2. Make sure you have the correct number of files in your "SavedSims" folder and then close Body Shop.

3. Move the files over to a new folder. To make things easier you can also copy the mesh files to that folder, that way you won't have to look for them later.

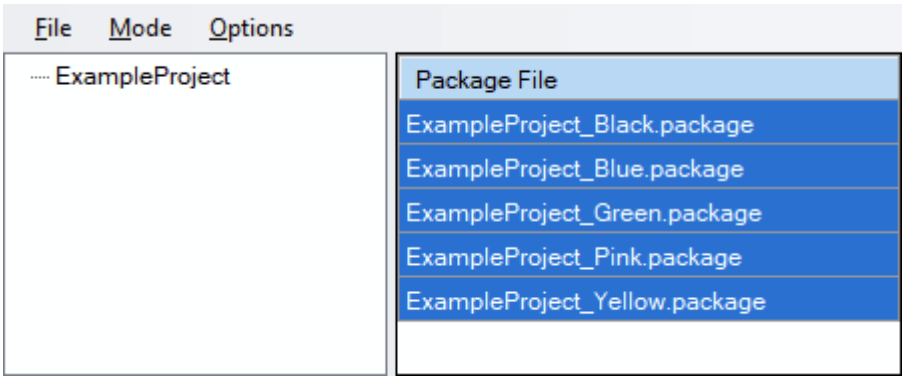
4. Rename all the files to something more intelligible. Remember that you can sort them by date to see which ones were created first, this will give you the order they appear in the catalog as well.



5. Open the RW. Go to **File > Select Folder** and select the folder you created in Step 3.



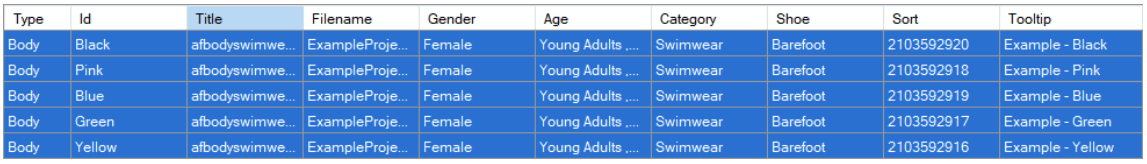
6. Select all the files that will be repositoried (in this case, all of them).



7. Give each item an ID (or not, depending on your naming conventions), and then select the items you want to repository.

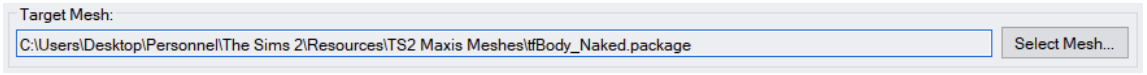
   Tip: You can use the "Tooltip" column (if you set them beforehand) or the "Filename"

column (mouse over it, or expand the column to see the full name) to help you identify each item. Alternatively, if you loaded the items in Body Shop once, and did not delete the "*cigen.package*" file afterward, mouseovering an item will display its thumbnail.



Type	Id	Title	Filename	Gender	Age	Category	Shoe	Sort	Tooltip
Body	Black	afbodyswimwe...	ExampleProje...	Female	Young Adults ,...	Swimwear	Barefoot	2103592920	Example - Black
Body	Pink	afbodyswimwe...	ExampleProje...	Female	Young Adults ,...	Swimwear	Barefoot	2103592918	Example - Pink
Body	Blue	afbodyswimwe...	ExampleProje...	Female	Young Adults ,...	Swimwear	Barefoot	2103592919	Example - Blue
Body	Green	afbodyswimwe...	ExampleProje...	Female	Young Adults ,...	Swimwear	Barefoot	2103592917	Example - Green
Body	Yellow	afbodyswimwe...	ExampleProje...	Female	Young Adults ,...	Swimwear	Barefoot	2103592916	Example - Yellow

8. Select a Target Mesh. This is the mesh that will be used by the items you are currently repositorying (eg. if the AF version is the master file, and TF the repo'd version, then here select the TF mesh).
   Keep in mind that subset numbers AND names must match the master files mesh!

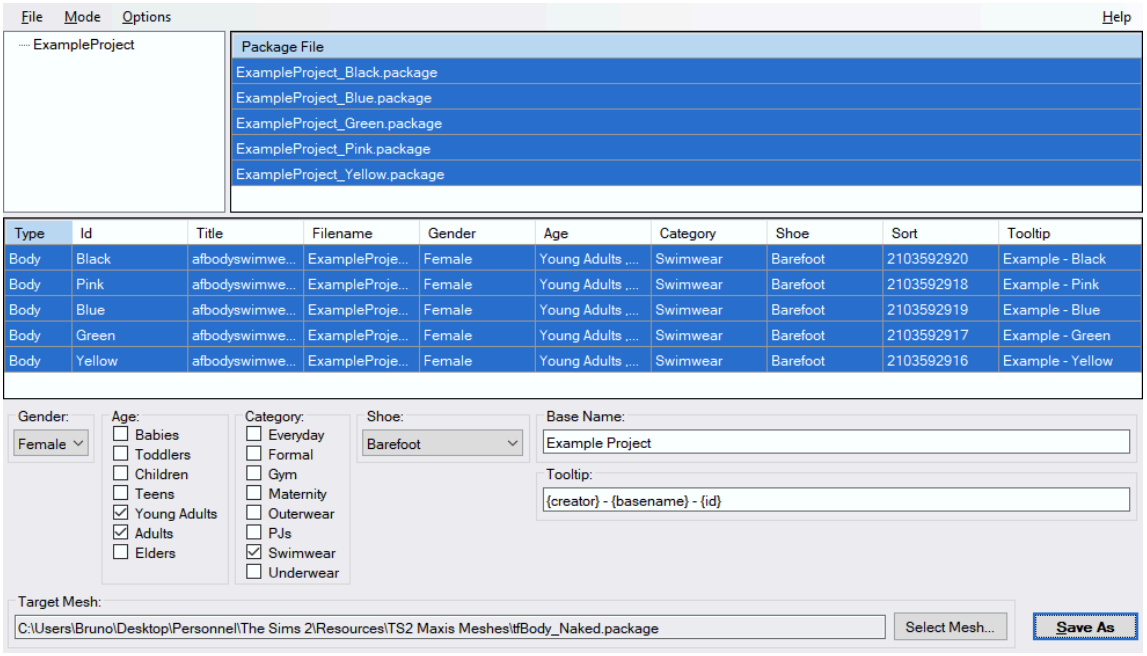


9. Change all the flags that apply to the repo'd items.

10. Enter a Base Name. **This step is required!**

11. Enter a Tooltip. Not required but **recommended**! (Learn more about it here)

12. When all that is done, you should have something like this:



13. Finally click the "Save As" button, and enter a name for your package file(s).

    a. If you choose **not** to merge the files, the RW will add the IDs typed in Step 7 as a suffix to the file's names to differentiate them.

    b. If no IDs were set, then the row index will be used as the default ID (counting starts at 1).

14. All going well, the saved files should each be around 1kb in size (if you merged them, multiply that by the number of repo'd items). They also should appear correctly in-game and Body Shop.

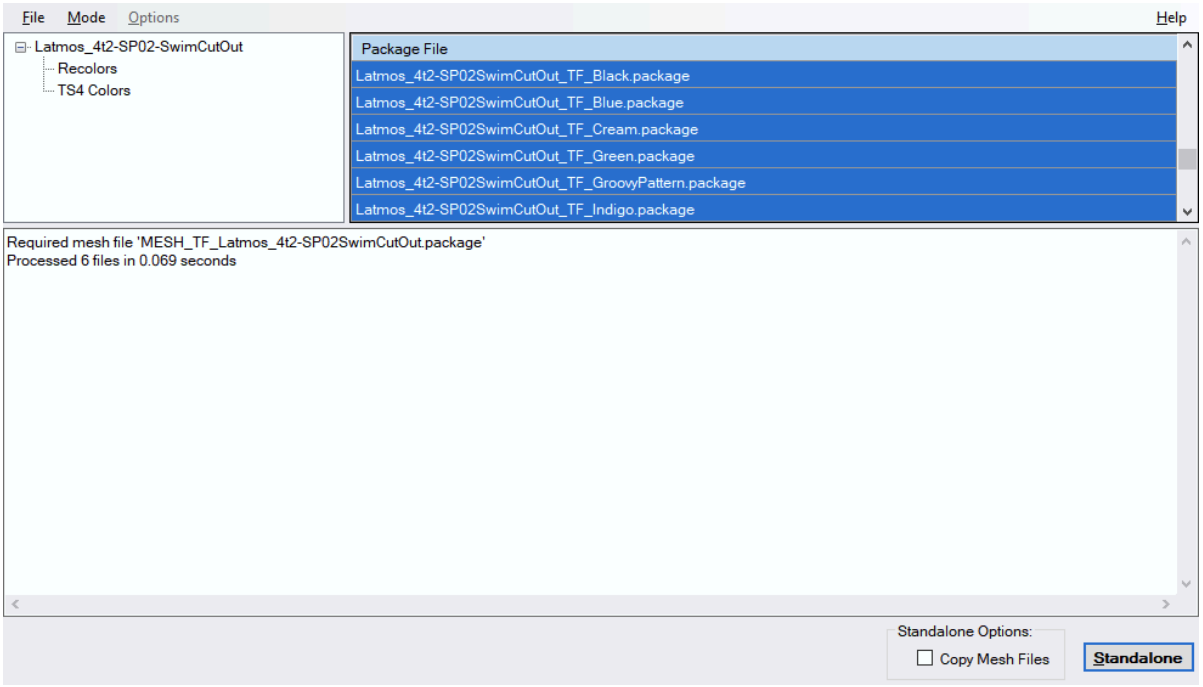
How to de-repository (make standalone)

The process is pretty straightforward:

1. Open the RW. Go to **File > Select Folder** and select the folder containing the files of items you wish to make standalone.
2. Select all the files you wish to make standalone.
3. If you wish to make a copy of the mesh files, check the corresponding option. **This option requires the mesh to be inside the folder currently loaded**. Doesn't matter if it's in the same folder/subfolder as the processed files or not, just needs to be somewhere in the loaded folder structure.
4. Click the "Standalone" button. **The files will be saved in a subfolder created by the RW**.

    a. If you see "Required mesh file (…)" or "No package file available for required mesh (…)" type of messages, worry not: the RW is just telling you it did or didn't find the mesh file for the processed items. **This doesn't impact the successful result** since the mesh isn't required in order for the de-repositorying to be done.

Tip: If you plan on uploading this as an alternative version to a repo'd one, the "Copy Mesh Files" option is useful to quickly set up a folder that can be uploaded separately.



More advanced

About tooltips

While it is not required to assign a new tooltip, leaving the field empty will result in the tooltip **being repo'd** to the master file. This means the repo'd item **WILL NOT** have a tooltip of its own, instead it pulls it from the master file (so no, it does not get copied over).

In the event that the master file is missing from the game, leaving matters of missing textures aside, the item will still function properly (no crash or bugs). But obviously it will have no tooltip!

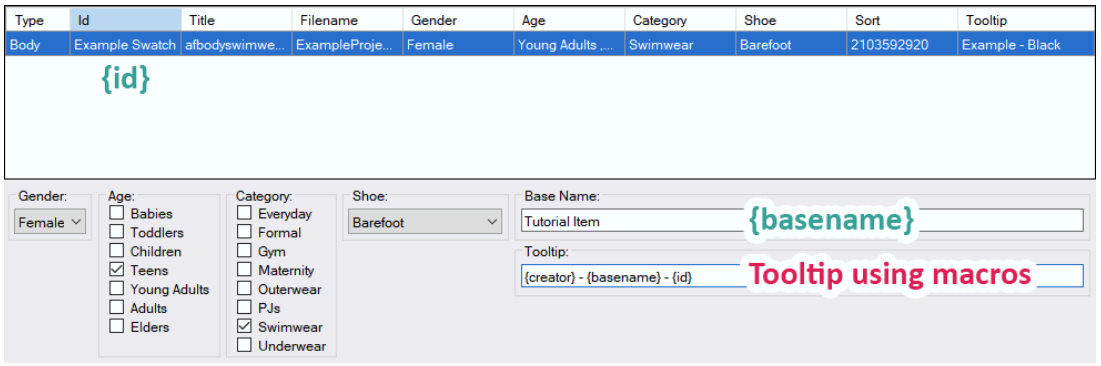
Note: Tooltips are saved in between sessions (app loads), as of v1.2. If you always use the same tooltip convention, this means you can set them up using Macros (see next section) and keep using that.

About macros

Macros can be used to fill in some info concerning the items automatically for you, without the need to type them individually or repeatedly. Here's the list of available macros:

`{creator}`	The creator's nickname (as configured in **File > Creator Details**).
`{basename}`	As set in the "Base Name" field.
`{age}`	As set in the "Age" flags, separated by a comma if multiple are selected.
`{agecode}`	Same as above but using Maxis age code conventions (B, P, C, T, YA/A, E).
`{gender}`	As set in the "Gender" flag (Female, Male or Unisex).
`{gender:N}`	Same as above but only keeps the amount of letters determined by "N". *For example:* if gender is unisex, {gender:1} will render "u".
`{type}`	The item's outfit part type (body, top or bottom).
`{id}`	The ID column value, or row index (starting at 1) if no ID was set.

This feature is mainly used for setting up tooltips quickly. Here's an example:



Using this configuration, my tooltip will render like this: Latmos - Tutorial Item - Example Swatch.

How to separate shared resources into their own files

If you're like me, and like to optimize your files as much as you can, while also trying to accommodate other people's preferences, separating shared resources might be something you'd wanna look into.

Note: This method produces more package files, which does contribute to longer loading times. However, since merging files in bulk is a thousand times easier than manually editing each one individually to clean up unwanted stuff, I feel like that's a good compromise to make.

Make sure you follow Steps 1 through 7 of the "How to repository" step-by-step. Then:

1. Select a Target Mesh. This time instead of selecting the mesh of the repo'd items, we are going to select the master files mesh. Get it? We are essentially repositorying the master files to themselves.
2. **Don't change any flags** this time around since those are already correct, **but do type in a Base Name**. This means the Property Sets will also be properly named this way, neat!
3. If you want your tooltips to be shared (I personally do), then leave the "Tooltip" field empty. If not, then like previously, set up your tooltips (don't forget about macros!).
4. Then go ahead and click "Save As" to export all the files.
5. Repeat this process for any items that will use the shared resources, making sure to change relevant settings (Target Mesh, Flags, Tooltip, …) for each version.
   Note: This step can also be done at the end, but you'll need to select already repo'd items instead of the original packages.
6. Now we need to clean up the original packages: in SimPE, open each file individually and delete any resource that isn't a TXMT, TXTR or (if you repo'd the tooltips) STR#.
    a. You can also use this time to import the textures through SimPE, since that would typically be done for each file one-by-one as well.
7. Rename the cleaned up packages if needed, and organize/label them clearly.
   Note: If people decide to delete some recolors for every version, the separated resources would also need deleting or else the game will load them for nothing.

How it works

Initially repositorying of Body Shop items was done by copy/paste-ing the master's TXTR resource name (e.g. ##0x12345678!outfit~stdMatBaseTextureName) into the repo'd file TXMT resource (which itself wasn't repo'd). Simple enough, but tedious to do in bulk.

What the RW does is directly alter the 3IDR, which contains almost all references to the item's other components (like the textures, material definitions, tooltips, …).

Simply put: it clones the master item's resources, assigns them new TGI values, updates the 3IDR with the new mesh references and changes any other stuff input by the user. And of course, allows this to be done in bulk for all selected items.

The end result is a repo'd item a bit lighter than the "manual" method, containing the standard "3IDR/BINX/GZPS" resource combo that is responsible for making the item appear correctly in game, anything else is repositoried. Also, depending on the user input, the STR# (tooltip) resource can be made standalone, or repo'd as well.

If the relation between resources inside a package file is something that interests you, there's a guide on Mod the Sims written by Echo that explains those connections in detail.

About "Verify SHPE Subsets" option

This next section only applies to custom Body Shop meshes. As of v1.2, this option is now unchecked by default in the RW.

When both "Verify SHPE Subsets" and "Verify GMDC Subsets" are active, the RW will check those two resources before it attempts to repository the items. This is because the RW requires that both the master AND slave meshes share the same number and names for their subsets.

So in theory, this option shouldn't be unchecked, right?

The problem is: a large number of custom meshes for Sim parts don't update the SHPE resource which normally contains a list of the subsets that the mesh possesses. While this doesn't seem to affect their functionality, it is considered better practice to update the SHPE resource to reflect any changes to the mesh's subsets in order to maintain the integrity of the Scenegraph.

However, letting the option stay checked will also result in the RW throwing an error for most meshes, telling you that the subsets don't match. Which is why it is now unchecked by default.