Under There!

What it is: Under There! is a Synthesis patcher that will distribute undergarments to NPCs in Skryim. These undergarments are imported from other mods (dependencies) and assigned to NPCs either at random or by rules. They will be worn by NPCs under clothes/armor, and will immediately be visible upon looting.

Why would you make this? As a VR player, I want realism in my game. I use realistic body textures (randomized via zEBD!) - I want a "Game of Thrones / Witcher" style of immersive fantasy game. What that means is that I don't want to leave a bunch of naked corpses behind while looting. Some tasteful nudity is great; a trail of naked corpses not so much. Also, I want to be able to show off my cool VR setup without worrying about how people will react to video game nudity. This mod aims to fix these issues by giving NPCs low-value undergarments that I can leave unlooted because I'm an adventurer, not a medieval panty snatcher.

How is this mod different from X? Underwear in Skyrim seems to be a saturated market, and there are two strategies to implement it without having it clipping through armor and clothes:

1. "Just in time" underwear. In this implementation, the NPCs receive underwear via script after their armor or clothes are looted. Examples of this approach are ws's new and great Equippable Underwear for NPCs and the older Smalls - (Just In Time Underwear).

   Advantages:
   - Mods can be relatively small - just armor, scripts, and the spells or effects to activate them.
   - Easy to install, few or no compatibility patches required

   Disadvantages:
   - Underwear distribution is done via Papyrus, which kicks in *after* the NPCs' clothes or armor is removed. Therefore, there is a flash of the body before it gets covered by underwear.

2. "At launch" distribution. In this implementation, the NPCs receive underwear by modifying their record in a .esp file. Because such a strategy would require compatibility patching with virtually everything, it is best-performed using a patcher that generates an esp from the user's load order (SSEedit, zEdit, Synthesis). The previous example of this approach is Equipable Underwear for Everyone, an SSEedit patcher.

   Advantages:
   - NPCs get the item at launch, so they're already wearing it when you loot them (their worn clothes or armor and also patched so as to not clip with the underwear). That means when you loot them, the underwear appears immediately, with no flash of a naked body.

- (Under There! only): additional items can be imported into the patcher and distributed to NPCs via editing the .json configuration file in a text editor (see the “Adding or Replacing Undergarments” section of the ReadMe or Manual).

Disadvantages:

- By nature of how Skyrim armors work, all other armors need to be patched to avoid clipping with the underwear if it is to be worn at the same time as their clothes. Therefore, every time your load order changes, you have to rerun the patcher to patch it accordingly. IMO it’s not that big of a burden, but it’s more cumbersome than just checking or unchecking a .esp file.

Under There! is a Synthesis patcher heavily inspired by Equipable Underwear for Everyone (in fact, the default settings expect you to have the EUfE installed so that it can make use of its underwear assets. It functions very similarly - you run the patch and it distributes underwear (either the default set or variants) to all humanoid NPCs. The key improvements over EUfE are:

1. Leveled NPCs that inherit their inventory from a template get the correct gendered items. Due to the way leveled lists are structured, the patchers can’t know in advance whether a leveled actor will be male or female. Therefore, the EUfE patcher distributes both tops and bottoms to all leveled actors, resulting in male guards, soldiers, bandits, etc having equipped (invisible) tops. While I have nothing against IRL crossdressers, this is not how I envision such characters behaving in the world of Skyrim, and it’s a bit immersion breaking in the loot inventory screen. Under There! makes use of a light script (distributed via powerofthree’s amazing Spell Perk Item Distributor) to fix gendered items, removing tops from male characters while leaving them on the females.

2. Under There! does not suffer from weird NPC skipping bugs (that seem to be caused by SSEedit bugs rather than EUfE itself). EUfE will skip over some NPCs for reasons that are not readily apparent. You can test this for yourself by trying to patch Immersive Patrols SE - EUfE will for some reason skip the soldiers while Under There! will patch them correctly.

3. Under There! is customizable via a json settings file (described in detail in the Configuration Section of the manual). You can import items from other plugins if you define them in the settings file and distribute them along with or instead of the default EUfE underwear. VR players, fear not the 255 plugin limit - after the patcher imports items, the source plugin(s) can be disabled so you don’t have to spend a precious esp slot just for an underwear mod.

Requirements:

- Spell Perk Item Distributor: **hard requirement** - to correctly edit gendered NPC inventory
- Equipable Underwear for Everyone: **soft requirement** - the default settings expect to import EUfE underwear, but I expect to make other settings files that merge or replace these items with those from other mods.

How to use:

- Install the patcher via Synthesis. Click “Git Repository” in the top left corner next to the + sign. If it is not visible in the Browse patcher list, click “Input” and paste in the patcher’s git repository URL: https://github.com/Piranha91/Under-There-. Wait a few seconds for it to find the patcher, click the bar under “Project” and select UnderThere\UnderThere.csproj. Wait another few seconds and click the checkbox that should turn blue in the bottom right corner. Then return to the main menu, click the circle next to Under-There- so that it turns into a checkbox, and click the Run (arrow) button at the bottom to run the patcher. It should only take a few seconds to complete.

Configuration

The settings for Under There! are defined in the .json file found in Synthesis\Data\Under-There-\UnderThereConfig.json. Open this file using your favorite json editor (I prefer VS Code).

The options in the json file are as follows:

AssignmentMode: Determines how underwear will be distributed. Can be set to:

- “default”: Only the default item(s) (defined below) will be distributed.
- “random”: All defined items will be distributed at random to any humanoid NPC
- “class”: Items will be distributed based on the NPC’s class combined with the **ClassDefinitions** defined below.
- “faction”: Items will be distributed based on the NPC’s faction combined with the **FactionDefinitions** defined below.
    - The default configuration is “faction” because this provides the most “precise” way to define NPC groups.

bPatchMales: If true, will give undergarments to male NPCs. If false, will not.

bPatchFemales: If true, will give undergarments to female NPCs. If false, will not.

bPatchNakedNPCs: If true, NPCs that are supposed to be naked in-game will be eligible to receive underwear. If false, they will not.

bPatchSummonedNPCs: If true, summoned NPCs will be eligible to receive underwear. If false, they will not. (Determined by the NPC’s “Summonable” flag).

bPatchGhosts: If true, ghost NPCs will be eligible to receive underwear. If false, they will not. (Determined via several methods because Bethesda didn’t use a single flag to define ghosts. Having “Ghost” in the name or EditorID is *not* one of these methods).

bMakeItemsEquippable: If true, underwear can be looted from NPCs just as any other armor. If false, the underwear will not be lootable (so you can use the “Remove All” key without taking the undergarments). However, they are still separate from the body model so if other mods force the NPC to remove all of its items, they will lose the underwear.

PatchableRaces: NPCs of these races are *always* eligible to be patched. Races not belonging to this list are still eligible to be patched provided that they also conform to a set of rules to filter out creatures, monsters, etc (this is because Guards and several other templated NPC types are set to non-humanoid races such as FoxRace, so only patching humanoid races would exclude these NPCs, which would be unfortunate because these are actually the most likely to be looted.

NonPatchableRaces: NPCs of these races are *never* eligible to be patched. This list is for races that are never inherited by templated humanoid NPCs (but are inherited by templated non-humanoid NPCs).

ClassDefinitions: A dictionary that defines which classes should receive which underwear sets (if **AssignmentMode** = “class”). Format is:

“**Label”**: [“class1”, “class2”, … “class(n)”].

By default there are three labels (“Poor”, “Medium”, and “Rich”). However, you can add or remove labels and lists if you want to have other categories. Categories must match to those listed in **Assignments** (with the exception of “Default”). If the NPC’s class is not found in the dictionary, it will be assigned the hardcoded “Default” item set.

FactionDefinitions: A dictionary that defines which factions should receive which underwear sets (if **AssignmentMode** = “faction”). Format is:

“**Label”**: [“faction1”, “faction2”, … “faction(n)”].

By default there are three labels (“Poor”, “Medium”, and “Rich”). However, you can add or remove labels and lists if you want to have other categories. Categories must match to those listed in **Assignments** (with the exception of “Default”). If an NPC belongs to multiple factions, they will all be counted and the label with the most matched factions will win (in the event of a tie, the winner will be chosen at random). If none of the NPC’s factions are found in the dictionary, it will be assigned the hardcoded “Default” item set.

FallBackFactionDefinitions: Same as **FactionDefinitions** but only used if none of the NPC’s factions are matched by any of the **FactionDefinitions**. For example, “JobJarlFaction” is by default in the “Rich” list of **FactionDefinitions**, and “TownWinterholdFaction” is in the “Medium” list of the **FallBackFactionDefinitions**. Korir belongs to both factions, but since “TownWinterholdFaction” is in the fallback list, he will get items from the “Rich” list. If **FallBackFactionDefinitions** is used, the winning label is selected in the same way as for **FactionDefinitions**.

IgnoreFactionsWhenScoring: If a faction is not found in the faction or fallback dictionary, Under There! will write a log notifying the user as such. This list simply tells the patcher not to log this faction because it's not supposed to influence label assignment.

SpecificNPCs: A list of NPCs to which the user can assign specific Labels or Item Sets.
Example:

```
"SpecificNPCs":
[
    {
        "Name": "Brenuin",
        "FormKey": "013BA7:Skyrim.esm",
        "Type": "group",
        "Assignment_Set": "Poor Variant 1",
        "Assignment_Group": "Rich"
    }
]
```

- "Name": purely for the user's information as a label for this SpecificNPC.
- "FormKey": The identifier of the NPC in the format of a Synthesis FormKey string (the last 6 digits of an NPC's FormID followed by a : and the NPC's source plugin (case-sensitive).
- "Type": Determines the type of object being assigned to the NPC. Can be "group" or "set".
- "Assignment_Set": if "Type" = "set", this must match one of the Set Names defined in **Sets**. The NPC will receive this specific item set.
- "Assignment_Group": if "Type" = "group", this must match one of the labels defined in **Assignments**. The NPC will receive one of the item sets corresponding to this Label.

"**BlockedNPCs"**: A list of NPCs that the user can prevent from being assigned underwear.

Example:

```
"BlockedNPCs":
[
    {
        "Name": "Jon BB",
        "FormKey": "013BB1:Skyrim.esm"
    }
]
```

- "Name": purely for the user's information as a label for this SpecificNPC.

- “FormKey”: The identifier of the NPC in the format of a Synthesis FormKey string (the last 6 digits of an NPC’s FormID followed by a : and the NPC’s source plugin (case-sensitive).

“**Assignments**”: A dictionary that maps item sets (lists of specific items) to the Labels referenced by the above settings.
Format: “Label”: [“item set name #1”, “item set name #2” …]
“**Sets”**: A list of item sets (referenced by the **Assignments**).

Each **Item Set** has the following “**UTset**” elements:

“Name”: The name of the item set (must match to that in **Assignments**).
“Items_Mutual”: Items (**UTitem** objects referencing ARMOR records) for this item set that are to be assigned to NPCs of both genders.
“Items_Male”: Items (**UTitem** objects referencing ARMOR records) for this item set that are to be assigned only to male NPCs.
“Items_Female”: Items (**UTitem** objects referencing ARMOR records) for this item set that are to be assigned only to female NPCs.

*Note: There must be at least one item for each gender; i.e. at least one item in **Items_Mutual** or, if no items are mutual, then at least one item in both **Items_Male** and **Items_Female**.*

Within each of the **Items_X** elements is a list of “**UTitem**” elements. Each **UTitem** has the following elements:

“Record”: The ARMOR record that is to be implemented as underwear in the format of a Synthesis FormKey string (the last 6 digits of an NPC’s FormID followed by a : and the NPC’s source plugin (case-sensitive).

“**DispName**”: The desired in-game name of this item.

“Weight”: The desired in-game weight of this item.

“**Value**”: The desired in-game value of this item.

Adding or Replacing Undergarments

Adding new undergarments is a relatively straightforward process with the following steps:

1. In the **Sets** section, copy one of the existing sets and paste it at the end of the list (separated from the previous Set by a comma).
2. Replace the **Name** with a new unique name for this set.

3. Edit the **Items_Mutual**, **Items_Male**, and **Items_Female** lists to reference the armor object(s) that you want to distribute in this set. If it is a brassiere, add it to **Items_Female**. If it is a bottom or unisex item, add it to **Items_Mutual**. If it is a male-only item of some sort, add it to **Items_Male**. Make sure that there is at least one armor referenced for each gender (i.e. if you're adding a female-only piece of clothing, add it to **Items_Female** and then pick the closest matching male equivalent armor that you can think of and add it to **Items_Male**.
4. In the **Assignments** section, add your **Set**'s name to one or more of the lists (or create a new list if you don't want to mix it with existing variants in the leveled lists). If you want your set to be the default set, replace "Default Variant" with the name of your new Set.
5. If you chose to add a new list to the **Assignments** section in step 4, add that list as a new list in **ClassDefinitions** (populated by the desired Classes) if distributing by NPC Class, or in **FactionDefinitions** and **FactionFallBackDefinitions** (populated by the desired Factions) if distributing by NPC faction.

Acknowledgements:

Noggog (Synthesis Discord channel) for enduring an endless amount of my questions as I was learning how to write Synthesis patchers.

EZ (Skyrim Modding Hub Discord channel) for a long discussion about how slots work.

Noah Boddie (Skyrim Modding Hub Discord channel) for Papyrus help.

firehazard (r/SkyrimMods Discord channel) for refactoring and cleaning up my Papyrus script.

SirJesto (r/SkyrimMods Discord channel) for Papyrus help.

wSkeever (r/SkyrimMods Discord channel) for general consultation.

Fuzzles (Synthesis Discord channel) for general encouragement.