

ReProccer Reborn Guide

A Guide to ReProcceR Reborn

(for v0.26+)

created by Ingvion

2025

Contents

Contents.....	2
Introduction.....	4
What it does.....	4
Rules.....	6
Blocks guide.....	7
Comments.....	8
Weapons patcher ("weapons").....	9
Weapon type overriding ("typeOverrides").....	10
Weapon type factors ("types").....	11
Weapon material overriding ("materialOverrides").....	12
Weapon material factors ("materials").....	13
Additional damage adjustments ("damageModifiers").....	14
Renaming ("renamer").....	15
Blacklisting ("excluded...").....	16
Armor patcher ("armor").....	17
Armor material overriding ("materialOverrides").....	18
Armor material factor ("materials").....	19
Additional armor adjustments ("armorModifiers").....	20
Adding factions keywords ("masquerade").....	21
Renaming ("renamer").....	22
Blacklisting ("excluded...").....	23
Alchemy patcher ("alchemy").....	24
Effects patching ("effects").....	25
Blacklisting ("excluded...").....	26
Projectiles/ammo patcher ("projectiles").....	27
Base stats ("baseStats").....	28
Material influence ("materialStats").....	29
Modifier influence ("modifierStats").....	30
Blacklisting ("excluded...").....	31
Errors.....	32
Unable to find the '_rules-basic.jsonc'.....	33
Unable to find the 'english.jsonc'.....	34
Unable to parse 'filename'.....	35
Has a rule but no material keyword.....	36
Unable to determine the material.....	37
Unable to determine weapon type.....	38
Unable to determine armor slot.....	39
No projectile.....	40
Appendix.....	41
Weapon materials map.....	42
Weapon types map.....	43

Armor materials map.....	45
Renamer.....	47
Localizing ReProcceer Reborn.....	49
General info.....	50
Language.....	51
Strings.....	52
Nouns with gr. gender.....	53
Gendered adjectives.....	54
Rules translation.....	55

Introduction

ReProccer Reborn is a port of [T3nd0's ReProccer](#) - the automated SkyRe patch generator - to zEdit's Unified Patching Framework. This ported patcher does everything the original ReProccer does to make armor, weapons, ammunition, and ingredients compatible with T3nd0's Skyrim Redone.

The reason both the original ReProccer and this port exists is to alleviate the need for hand-made compatibility patches for armor, weapon, ammunition, and ingredients added by mods. The ReProccer also allows users to quickly adjust how weapons and armor are balanced by editing rules.

What it does



Weapons:

- Applies new or redefines existing type and material keywords on vanilla and mod-added weapons, based on the rules
- Modifies any weapon stats according to its type and material
- Modifies tempering recipes according to weapon materials
- Generates breakdown recipes for playable weapons
- Generates SkyRe enhanced crossbow versions of vanilla and mod-added crossbows, along with crafting, tempering, and breakdown recipes for them
- Generates Refined Silver versions of playable vanilla and mod-added silver weapons, along with crafting, tempering, and breakdown recipes for them
- Renames any weapon according to the rules



Armors and clothing:

- Applies new or redefines existing material keywords on vanilla and mod-added armors, based on the rules
- Modifies any armor stats according to its weight and material
- Classifies shields as light or heavy (based on material) by renaming and distributing relevant keywords

- Sets the new cap for maximum armor protection and adjusts the armor protection algorithm
- Distributes [Masquerade](#) perk keywords on vanilla and mod-added armors
- Adds crafting requirements to leather armor recipes
- Modifies tempering recipes according to armor materials
- Generates [Dreamcloth](#) versions of playable vanilla and mod-added clothing, along with crafting and breakdown recipes for them
- Generates breakdown recipes for playable armors
- Renames any armor, clothing, or jewelry according to the rules

Ammunition:

- Modifies speed, damage, and gravity of ammo, based on their type and material
- Modifies weight and gold value of ammo, based on their type and material
- Generates SkyRe ammunition variants for vanilla and mod-added ammo, along with crafting recipes.

Ingredients:

- Makes most ingredients effects work over time
- Normalize extreme values for ingredients
- Sets min and max gold value of any ingredient

Rules

All ReProccer Reborn rules can be found as JSON or JSONC files at:


<zEdit folder>\modules\reproccerReborn\data

JSON/JSONC is a human readable format to store data. It's easy to read, understand and edit – any text editor will do, though I recommend the one with a syntax highlight feature, like [Notepad++](#).

JSON and JSONC are the same text-based format, but JSONC supports single line and multiline comments.

Rules files have the following hierarchy:

- **_rules-basic.jsonc** contains all basic rules, and has the lowest priority. This file should always be present in the rules folder.
- **_rules-user.jsonc** completes the chain of rules and contains no rules by default; it exists for user-defined rules, and has the highest priority.
- Other **.json/.jsonc** files are rules for eponymous mods; they override rules defined in **_rules-basic.jsonc** or define their own. These rules priority is the same as mods load order.

 Rules order in files is also important - next in a list have higher precedence, just like plugins in load order.

Blocks guide

This section explains what each rules block is responsible for.

Note that with 0.25 the basic rules file **_rules-basic.jsonc** contains most of the info below, so you can always study the rules in vitro instead of reading this part of the guide.

Comments

As of 0.25, rules of .jsonc format support single line and multiline comments.

```
// I am a single line comment
// SL comments start with double slash
// SL comments cannot be wrapped to the next line
// This line is a correct SL comment, but
this line will cause a parsing error
```

Weapons patcher (“weapons”)

New damage value for a weapon is calculated as:

(base damage + material factor + type factor) * modifier


- **Base damage** is the damage value a weapon gets from its archetype - one-handed, two-handed, bow, crossbow. Configure these values on the ReProcceer Reborn settings page.
- **Material factor** is the damage value a weapon gets from its material, according to the [material factors list](#). Damage values associated with materials could be changed by editing rules files.
- **Type factor** is the damage value weapon gets from its type (waraxe, katana, shortsword etc), according to the [type factors list](#). Damage values with types could be changed by editing rules files.
- **Modifier** is the value on which the sum of base damage and factors is multiplied, based on a relevant rule for the weapon.

Weapon type overriding (“typeOverrides”)

A weapon with **names** in its name will have its type changed to a specified **type**.

You can change a weapon type by specifying the weapon name as **names**, and new type as **type**; ReProccer will remove the existing type keyword (if any), and assign the new one.


```
"typeOverrides": [
  { "names": "Bloodscythe",          "type": "Scimitar"  },
  { "names": ["Blade", "Pale"],      "type": "Longsword" }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **Valid weapon types** are:

Bastard, Battleaxe, Battlestaff, Broadsword, Club, Crossbow, Dagger, Glaive, Greatsword, Halberd, Hatchet, Katana, Longbow, Longmace, Longspear, Longsword, Mace, Maul, Nodachi, Scimitar, Shortbow, Shortspear, Shortsword, Tanto, Wakizashi, Waraxe, Warhammer

 Weapons with reassigned types will have their names in “**name [type]**” format in game (toggleable in the patcher’s options).

Weapon type factors (“types”)


A weapon with **names** in its name will have the corresponding **damage** as the **type factor**, and its reach and speed changed to **reach** and **speed**.

If ReProccer fails to determine the weapon type by the name, it will check the keywords for an **id** keyword, and will use a rule for the found one.

```
"types": [
  { "names": "Dagger",      "reach": 0.7, "speed": 1.3, "damage": 1, "id": "type_dagger" },
  { "names": "Longsword",  "reach": 1.1, "speed": 1.05, "damage": 4, "id": "type_longsword" },
  { "names": "Waraxe",     "reach": 0.95, "speed": 0.95, "damage": 8, "id": "type_waraxe" }
]
```


A weapon can have its type stats overridden with a respective rule for a name placed lower in the rules list.

```
"types": [
  { "names": "Zephyr",      "speed": 0.4 },
  { "names": "Saber",      "speed": 0.95, "damage": 5 },
  { "names": "Dawnbreaker", "reach": 1.12 }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **reach** and **speed** must be positive floats, **damage** must be an integer.

 Do not modify **id** values as they point to relevant values in the internal map of weapon types.


Weapon material overriding (“materialOverrides”)

A weapon with **names** in its name will have its material changed to a specified **material**.

You can change a weapon material by specifying the weapon name as **names**, and new material as **material**; ReProcceer will remove the existing material keyword (if any), and assign the new one.

Tempering recipe, if any, will be modified in accordance with the new material; breakdown recipe will be generated using the new material. The crafting recipe will not be modified.

```
"materialOverrides": [
  { "names": "Bolar's Oathblade",      "material": "Blades" },
  { "names": ["Ayleid", "Honed"],      "material": "Elven"  }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **Valid materials** are:

Amber, Dark, Golden, Madness¹, Blades, Daedric, Dawnguard, Dragonbone, Draugr, Draugr Honed, Dwarven, Ebony, Elven, Falmer, Falmer Honed, Forsworn, Glass, Imperial, Iron, Nordic, Orcish, Silver, Stalhrim, Steel, Wood

¹ These materials originate from the “Saints & Seducers” CC mod, and cannot be assigned without it.

Weapon material factors (“materials”)


A weapon with **names** in its name will have the corresponding **damage** as the **material factor**.

If specified, **speedMod** will be added to the weapon type's base speed for any weapon of this material.


If specified, **critMult** will override the default critical damage multiplier (x0.5) for any weapon of this material.

If ReProccer fails to determine the material by the name, it will check the keywords for an **id** keyword, and will use a rule for the found one.

```
"materials": [
  { "names": "Blades",    "damage": 5,    "speedMod": -0.1,    "id": "mat_blades" },
  { "names": "Daedric",   "damage": 9,    "id": "mat_daedric" },
  { "names": "Silver",    "damage": 2,    "critMult": 0.05,    "id": "mat_silver" },
  { "names": "Ash",       "damage": 2,    "speedMod": -0.03    },
  { "names": "Draugr",    "damage": 1    }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **damage** must be an integer, **speedMod** must be a float, **critMult** must be a positive float.


 Do not modify **id** values as they point to relevant values in the internal map of materials.

Additional damage adjustments (“damageModifiers”)

A weapon with **names** in its name will have the corresponding value as the **modifier**.

Note that multiplier only affects the weapons’ physical damage; any damage-dealing enchantments are unaffected.

```
"damageModifiers": [
  { "names": "Dragonbane", "multiplier": 1.1 },
  { "names": ["Dwarven", "Ballista"], "multiplier": 1.4 }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **multiplier** must be a positive float.

Renaming (“renamer”)

Search for the **find** in the name and replace it with the **replace**.

If specified, **filter** restricts a rule it's set for to weapons with the certain animation type only; **options** determine search and replace modes.

```
"renamer": [
  { "find": "Sword", "replace": "Broadsword", "options": "ic", "filter": "OneHandSword" },
  { "find": "Bow", "replace": "Shortbow", "options": "ic", "filter": "Bow" }
]
```

⚠ If all words from the **replace** already present in the weapon name (in any order), renaming will not be performed.

⚠ **options** must contain any combination of the following values:

- **i** make search case-insensitive
- **g** search and replace all instances of the substring
- **p** allow substring as the part of a word
- **c** retain original case
- **n** search for the next rule
- **o** allow weapons with overridden type

See [examples](#) for a better understanding of how flags work.

⚠ **filter** must contain one of the following values:

Bow, Crossbow, OneHandSword, OneHandDagger, OneHandAxe,
OneHandMace, TwoHandSword, TwoHandAxe


Set the **filter** to "" to check weapons regardless of their animation type (not recommended).

Blacklisting (“excluded...”)

A weapon that has at least one of the listed strings in its name will be excluded from a relevant patching process.

- ❖ “excludedFromRenaming” - no renaming
- ❖ “excludedFromRecipes” - no recipe modification
- ❖ “excludedSilver” - no Refined Silver variants
- ❖ “excludedCrossbows” - no SkyRe crossbow variants
- ❖ “excludedWeapons” - full exclusion

```
"excludedFromRenaming": [],
"excludedFromRecipes": [],
"excludedFromSilver":  ["Bloodskal Blade"],
"excludedCrossbows":   ["Enhanced"],
"excludedWeapons":      ["Dummy", "FXdustDropWEP"]
```

 If exclusion by editor ID is allowed (in patcher settings), **excludedWeapons** additionally checks if the editor ID fully match any of the strings, and excludes the weapon if so.

Only **excludedWeapons** can check for editor ID match.

Armor patcher (“armor”)

New armor value for any armor is calculated as:

$$(\text{material factor} * \text{slot factor}) * \text{modifier}$$

- **Material factor** is the armor value an armor gets from its material, according to the [material factors list](#). Armor values associated with materials could be changed by editing rules files.
- **Slot factor** is the armor value multiplier an armor gets from its equip slot (cuirass, boots, helmet, etc). A multiplier for each slot can be configured on the ReProccer Reborn settings page.
- **Modifier** is a value on which the result of multiplying material factor and slot factor will be additionally multiplied, based on relevant rules for an armor.


Armor material overriding (“materialOverrides”)

An armor with **names** in its name will have its material changed to a specified **material**.

You can change an armor material by specifying the armor name as **names**, and new material as **material**; ReProccer will remove the existing material keyword (if any), and assign the new one.

Tempering recipe, if any, will be modified in accordance with the new material; breakdown recipe will be generated using the new material. The crafting recipe will not be modified.

```
"materialOverrides": [
  { "names": "Morag Tong",          "material": "Chitin" },
  { "names": "Penitus Oculatus",    "material": "Steel"  }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **Valid materials** are:

Amber, **Dark**, **Golden**, **Madness**², Ancient Nord, Blades, Bonemold, Chitin Heavy, Chitin, Daedric, Dawnguard, Dragonplate, Dragonscale, Dwarven, Ebony, Elven, Elven Gilded, Falmer, Falmer Hardened, Falmer Heavy, Forsworn, Glass, Guard, Hide, Imperial Light, Imperial Studded, Imperial Heavy, Iron, Iron Banded, Leather, Nightingale, Nordic, Orcish, Scaled, Shrouded, Stalhrim Heavy, Stalhrim Light, Steel, Steel Plate, Stormcloak, Stormcloak Officer, Studded, Thieves Guild, Vampire, Wolf


² These materials originate from the “Saints & Seducers” CC mod, and cannot be assigned without it.

Armor material factor (“materials”)


An armor with **names** in its name will have the corresponding **armor** as the **material factor**.

If ReProccer fails to determine the material by the name, it will check the keywords for an **id** keyword, and will use a rule for the found one.

```
"materials": [
  { "names": "Amber", "armor": 36, "id": "mat_amber"
},
  { "names": ["Gilded", "Elven"], "armor": 25, "id": "mat_elveng"
},
  { "names": "Deathbrand", "armor": 34
}
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.


Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **armor** should be a positive integer.

Additional armor adjustments (“armorModifiers”)

An armor with **names** in its name will have the corresponding value as the **modifier**.

```
"armorModifiers": [  
  { "names": ["Bonemold", "Improved"], "multiplier": 1.1 },  
  { "names": "Old Gods", "multiplier": 1.25 }  
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.


 **multiplier** must be a positive float.

Adding factions keywords (“masquerade”)


An armor with **names** in its name will get the corresponding **faction** keyword for the **Masquerade** perk.

If specified, **filter** restricts a rule it's set for to armors with listed armor types.


```
"masquerade": [
  { "names": "Old Gods", "faction": "Forsworn" },
  { "names": "Thalmor", "faction": "Thalmor", "filter": "Clothing, Light Armor" }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.


Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **filter** must contain any combination of the following values:

Clothing, Light Armor, Heavy Armor

 **faction** must contain one of the following values:

Bandit, Forsworn, Imperial, Stormcloak, Thalmor

 Faction keywords cannot be assigned to jewelry.

Renaming (“renamer”)

Search for the **find** in the name and replace it with the **replace**.

If specified, **filter** restricts a rule it's set for to armors of a certain weight class only; **options** determine search and replace modes.

If specified, **skipIf** does not perform renaming if any listed word already presents in the name.

```
"renamer": [
  { "find": "Ring", "replace": "Band", "options": "ic", "filter": "Clothing" },
  { "find": "Helmet", "replace": "Cap", "options": "ic", "filter": "Heavy Armor", "skipIf": "Blades" }
]
```

⚠ If all words from the **replace** already present in the weapon name (in any order), renaming will not be performed.

⚠ **options** must contain any combination of the following values:

- **i** make search case-insensitive
- **g** search and replace all instances of the substring
- **p** allow substring as the part of a word
- **c** retain original case
- **n** search for the next rule

See [examples](#) for a better understanding of how flags work.

⚠ **filter** must contain one of the following values:

Clothing, Light Armor, Heavy Armor


Set the **filter** to "" to check armors regardless of their weight class (not recommended).

Blacklisting (“excluded...”)

An armor that has at least one of the listed strings in its name will be excluded from a relevant patching process.

- ❖ “excludedFromRenaming” - no renaming
- ❖ “excludedFromRecipes” - no recipe modification
- ❖ “excludedDreamcloth” - no Dreamcloth variants
- ❖ “excludedArmor” - full exclusion

```
"excludedFromRenaming": [],
"excludedFromRecipes": [],
"excludedDreamcloth":   ["Ragged", "Skaal"],
"excludedArmor":        ["DLC01AurielsShieldTEST"]
```

 If exclusion by editor ID is allowed (in patcher settings), **excludedArmor** additionally checks if the editor ID fully match any of the strings, and excludes the armor if so.

Only **excludedArmor** can check for editor ID match.

Alchemy patcher (“alchemy”)

New magnitude for an ingredient effect is calculated as:

$$\text{default mag} * \text{mag factor}$$


- **Default mag** is the default magnitude of an ingredient effect; taken from the master file or last mod that changed this value.
- **Mag factor** is the value on which the default magnitude will be multiplied for each effect type (restore, regen, resist, etc).

New duration for any ingredient effect is equal to a specified in the rules for each effect type (restore, regen, resist, etc).


Effects patching (“effects”)

An effect with **names** in its name will get the corresponding **magnitudeMult** as the **magnitude factor**, and have its duration changed to the specified **duration**.

```
"effects": [
  { "names": "Damage",      "magnitudeMult": 0.75,    "duration": 5
  },
  { "names": "Fortify",     "magnitudeMult": 1.5,      "duration": 30
  },
  { "names": "Resist",      "magnitudeMult": 2,        "duration": 30
  }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.


 **magnitudeMult** must be a positive float, **duration** must be a positive integer.

Blacklisting (“excluded...”)

An ingredient effect, or an ingredient itself that has at least one of the listed strings in its name will be excluded from a relevant patching process.

- ❖ “excludedEffects” - no effects patching
 - ❖ “excludedIngredients” - full exclusion
-

```
"excludedEffects":      ["Damage Health"],  
"excludedIngredients": ["Jazbay Grapes"]
```

 If exclusion by editor ID is allowed (in patcher settings), **excludedIngredients** additionally checks if the editor ID fully match any of the strings, and excludes the ingredient if so.

Only **excludedIngredients** can check for editor ID match.

Projectiles/ammo³ patcher (“projectiles”)

New stats for any ammo are calculated as:

New stats:

(base stats + material stats) + modifier stats

New weight and gold value:

weight * modifier

gold value * modifier


- **Base stats** are projectile’s gravity, speed and damage.
- **Material stats** is an influence of the material on each base stat.
- **Modifier stats** is an influence of the modifier on each base stat.

³ Ammo are arrow and bolt items; projectiles are “shooting” arrows and bolts.


Base stats (“baseStats”)


An ammo with **names** in its name will get the corresponding **damage**, **gravity** and **speed** as the eponymous **base stats**.

```
"baseStats": [
  { "names": "Arrow", "gravity": 0.25, "speed": 5200, "damage": 7 },
  { "names": "Bolt", "gravity": 0.2, "speed": 6500, "damage": 13 }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **gravity** determines how fast the projectile loses momentum (higher value - faster loss of momentum); must be a positive float.

 **speed** determines how fast the projectile moves, in game units per second (1 unit = 0.5625"); must be a positive integer.


 **damage** must be a positive integer.

Material influence (“materialStats”)

An ammo with **names** in its name or in the name of its projectile will get the corresponding **gravity**, **speed** and **damage** added to the eponymous base values.


If ReProccer fails to determine the material by the name, it will check the keywords for an **id** keyword, and will use a rule for the found one.

```
"materialStats": [
  { "names": "Daedric", "gravity": 0.08, "speed": -700, "damage": 18, "id": "mat_daedric" },
  { "names": "Ebony", "gravity": 0.06, "speed": -550, "damage": 15, "id": "mat_ebony" },
  { "names": "Riekling", "gravity": 0.3, "speed": -1200, "damage": 10 }
]
```

 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

 **gravity** must be a float, **speed** and **damage** must be integers.

 Do not modify **id** values as they point to relevant values in the internal map of materials.

Modifier influence (“modifierStats”)

An ammo with **names** in its name will get the corresponding **gravity**, **speed** and **damage** added to the eponymous base values.

If specified, **weightMult** and **valueMult** will be multiplied by the default weight and value respectively.

```
"modifierStats": [
  { "names": "Hardened", "gravity": 0, "speed": 0, "damage": 6, "weightMult": 0.9, "valueMult": 1.5 },
  { "names": "Ashen", "gravity": 0.05, "speed": -200, "damage": -4, "weightMult": 1.1, "valueMult": 1.3 }
]
```

⚠ Keep in mind that ammo variants are **HARDCODED**; it's safe to change the values, but do not rearrange, add or remove rules here.

⚠ ReProccer generates ammo variants using already patched ammo as a template (i.e. with material influence).

📌 **names** contains full words or phrases that should be present in the name for the rule to be applied. Word case is irrelevant.

Specify several **names** for a rule by using the [brackets notation]; each should be present in the name in any order for the rule to be applied.

📌 **gravity** must be a float, **speed** and **damage** must be integers.


📌 **weightMult** and **valueMult** must be positive floats.

Blacklisting (“excluded...”)

An ammo that has at least one of the listed strings in its name will be excluded from a relevant patching process.

- ❖ “excludedAmmunitionVariants” - no ammo variants
- ❖ “excludedAmmunition” - full exclusion

```
"excludedAmmunitionVariants": ["Bound"],
"excludedAmmunition":         ["Test Bolt"]
```

 If exclusion by editor ID is allowed (in patcher settings), **excludedAmmunition** additionally checks if the editor ID fully match any of the strings, and excludes the ammo if so.

Only **excludedAmmunition** can check for editor ID match.

Errors

Here is the small guide on possible errors, and how to deal with them.

You can ignore any errors (unless they terminate the patching process), and use the ReProccer patch as is, however I do recommend fixing them - in most cases error means the affected record will not have intended values, and this could affect your playing experience.

Thanks to zEdit caching, ReProccer could be safely rebuilt mid-game, without the risk to lose ReProccer-originated items.

Keep in mind that errors on non-playable records can be safely ignored, because in general non-playables are props that intend to remain in the state they are; fix those only if you know what you're doing.

Unable to find the ‘_rules-basic.jsonc’

Error: Unable to find ‘_rules-basic.jsonc’

? Cause

The basic rules file **_rules-basic.jsonc** was not found in the **reproccerReborn/data** folder.

🔨 Solution

Make sure the **_rules-basic.jsonc** is present in the **reproccerReborn/data** folder.

Unable to find the 'english.jsonc'

Error: Unable to find 'english.jsonc'

? Cause

Default strings file **english.jsonc** was not found in the **reproccerReborn/locales** folder.

🔧 Solution

Make sure the **english.jsonc** is present in the **reproccerReborn/locales** folder.

Unable to parse '*filename*'

```
Error: Unable to parse 'filename'  
SyntaxError: error description
```


? Cause


A syntax mistake was made in a specified JSON/JSONC file (misplaced comma, unclosed brace, etc).

🔨 Solution

Double check the position indicated in the error description, or just let any online JSON5 validator, like [this one](#), do it for you. You can also use the JSON Validator script for zEdit from the ReProccer mod page.

Has a rule but no material keyword

→  Tower Steel Shield (0x123456):
has a rule for the name, but no material keyword

→  Big Elven Battleaxe (0x654321):
has a rule for the name, but no material keyword

? Cause

The record has no recognized material keyword, and cannot have its tempering recipe modified or breakdown recipe generated for it; however, a rule was found for the record name, and proper material factor for new damage calc formula was determined.

Solutions

- Ignore it - some mods intentionally do not use vanilla material keywords. Such records are not affected by some keywords-based perks, but will be patched anyway as long as there are rules for their names.
- Assign a material keyword by creating a rule ([for weapon](#)/[for armor](#)).

Unable to determine the material

- ❌ Bone Shield (*0x123abc*):
unable to determine armor material
 - ❌ Ayleid War Axe (*0x321fed*):
unable to determine weapon material
-


? Cause

The record has no recognized material keyword and there's no rule for the name.

🔨 Solutions

Assign a material keyword by creating a rule ([for weapon](#)/[for armor](#)), or create a rule for the name ([for weapon](#)/[for armor](#)).

Unable to determine weapon type

→  Demonic Pike (0x000666):
unable to determine weapon type


Cause

The weapon has no recognized type keyword.

Solutions

Assign a type keyword by [creating a rule](#).

Unable to determine armor slot

→  Elven Cape (0x112233):
unable to determine armor slot

Cause


The armor does not have a keyword determining its slot (boots, cuirass, etc).

Solution

Unless the armor piece is non-playable, lack of slot keyword indicates an error during the armor record creation.

There's no way to fix it from the ReProccer side; armors with this error will not have their armor value modified, so you can either ignore them, or add to the exclusion list to forbid ReProccer from attempts to patch them.

No projectile

→  Javelin (0x224466): has no projectile

Cause

The ammo does not have a projectile record attached to it.

Solution

Unless the ammo is a non-playable props, lack of projectile attached makes it unusable for its intended purpose.

There's no way to fix it from the ReProccer side; ammo with no projectile will not be patched.

Appendix

Weapon materials map

Name	Keyword	Master
Amber	ccBGSSSE025_WeapMaterialAmber	ccBGSSSE025-AdvDSGS.esm
Blades	WAF_WeapMaterialBlades	Update.esm *
Daedric	WeapMaterialDaedric	Skyrim.esm
Dark	ccBGSSSE025_WeapMaterialDark	ccBGSSSE025-AdvDSGS.esm
Dawnguard	WAF_DLC1WeapMaterialDawnguard	Update.esm *
Dragonbone	WeapMaterialDragonbone	Dawnguard.esm
Draugr	WeapMaterialDraugr	Skyrim.esm
Draugr Honed	WeapMaterialDraugrHoned	Skyrim.esm
Dwarven	WeapMaterialDwarven	Skyrim.esm
Ebony	WeapMaterialEbony	Skyrim.esm
Elven	WeapMaterialElven	Skyrim.esm
Falmer	WeapMaterialFalmer	Skyrim.esm
Falmer Honed	WeapMaterialFalmerHoned	Skyrim.esm
Forsworn	WAF_WeapMaterialForsworn	Update.esm *
Glass	WeapMaterialGlass	Skyrim.esm
Golden	ccBGSSSE025_WeapMaterialGolden	ccBGSSSE025-AdvDSGS.esm
Imperial	WeapMaterialImperial	Skyrim.esm
Iron	WeapMaterialIron	Skyrim.esm
Madness	ccBGSSSE025_WeapMaterialMadness	ccBGSSSE025-AdvDSGS.esm
Nordic	DLC2WeapMaterialNordic	Dragonborn.esm
Orcish	WeapMaterialOrcish	Skyrim.esm
Silver	WeapMaterialSilver	Skyrim.esm
Stalhrim	DLC2WeaponMaterialStalhrim	Dragonborn.esm
Steel	WeapMaterialSteel	Skyrim.esm
Wood	WeapMaterialWood	Skyrim.esm

* the keyword is injected in the Update.esm and does not exists in this file directly

Weapon types map

Name	Keyword	Master
Battleaxe	WeapTypeBattleaxe	Skyrim.esm
Glaive	skyre__WeapTypeGlaive	Skyrim AE Redone - Core.esm
Halberd	skyre__WeapTypeHalberd	Skyrim AE Redone - Core.esm
Longspear	skyre__WeapTypeLongspear	Skyrim AE Redone - Core.esm
Bow	WeapTypeBow	Skyrim.esm
Crossbow	skyre__WeapTypeCrossbow	Skyrim AE Redone - Core.esm
Longbow	skyre__WeapTypeLongbow	Skyrim AE Redone - Core.esm
(Broad)sword	WeapTypeSword	Skyrim.esm
Katana	skyre__WeapTypeKatana	Skyrim AE Redone - Core.esm
Longsword	skyre__WeapTypeLongsword	Skyrim AE Redone - Core.esm
Scimitar	skyre__WeapTypeScimitar	Skyrim AE Redone - Core.esm
Shortsword	skyre__WeapTypeShortsword	Skyrim AE Redone - Core.esm
Wakizashi	skyre__WeapTypeWakizashi	Skyrim AE Redone - Core.esm
Dagger	WeapTypeDagger	Skyrim.esm
Tanto	skyre__WeapTypeTanto	Skyrim AE Redone - Core.esm
Greatsword	WeapTypeGreatsword	Skyrim.esm
Bastard	skyre__WeapTypeBastard	Skyrim AE Redone - Core.esm
Nodachi	skyre__WeapTypeNodachi	Skyrim AE Redone - Core.esm
Mace	WeapTypeMace	Skyrim.esm
Club	skyre__WeapTypeClub	Skyrim AE Redone - Core.esm
Maul	skyre__WeapTypeMaul	Skyrim AE Redone - Core.esm
Waraxe	WeapTypeWaraxe	Skyrim.esm
Hatchet	skyre__WeapTypeHatchet	Skyrim AE Redone - Core.esm
Shortspear	skyre__WeapTypeShortspear	Skyrim AE Redone - Core.esm
Warhammer	WeapTypeWarhammer	Skyrim.esm
Quarterstaff	skyre__WeapTypeQuarterstaff	Skyrim AE Redone - Core.esm
Longmace	skyre__WeapTypeLongmace	Skyrim AE Redone - Core.esm

☞ Broadsword subtype is the same as the vanilla sword type.

☞ Shortbow subtype is the same as the vanilla bow type.

☞ Each weapon always has at least a keyword of its main type; a weapon with subtype has the subtype keyword, and the main type keyword.

☞ In the vanilla game crossbows do not have a separate type keyword, and are considered as bows.

Armor materials map

Light Materials		
Name	Keyword	Master
Amber	ccBGSSSE025_ArmorMaterialAmber	ccBGSSSE025-AdvDSGS.esm
Bonemold	DLC2ArmorMaterialBonemoldLight	Dragonborn.esm
Chitin	DLC2ArmorMaterialChitinLight	Dragonborn.esm
Dark	ccBGSSSE025_ArmorMaterialDark	ccBGSSSE025-AdvDSGS.esm
Dragonscale	ArmorMaterialDragonscale	Skyrim.esm
Elven	ArmorMaterialElven	Skyrim.esm
Elven Gilded	ArmorMaterialElvenGilded	Skyrim.esm
Forsworn	ArmorMaterialForsworn	Update.esm
Glass	ArmorMaterialGlass	Skyrim.esm
Guard	WAF_ArmorMaterialGuard	Update.esm *
Hide	ArmorMaterialHide	Skyrim.esm
Imperial Light	ArmorMaterialImperialLight	Skyrim.esm
Imperial Studded	ArmorMaterialImperialStudded	Skyrim.esm
Leather	ArmorMaterialLeather	Skyrim.esm
Nightingale	ArmorMaterialNightingale	Skyrim.esm
Scaled	ArmorMaterialScaled	Skyrim.esm
Shrouded	ArmorMaterialDarkBrotherhood	Skyrim.esm
Stalhrim Light	ArmorMaterialStalhrimLight	Dragonborn.esm
Stormcloak	ArmorMaterialStormcloak	Skyrim.esm
Stormcloak Officer	ArmorMaterialBearStormcloak	Update.esm
Studded	ArmorMaterialStudded	Skyrim.esm
Thieves Guild	ArmorMaterialThievesGuild	Update.esm
Vampire	kwArmorMaterialVampire	Dawnguard.esm

* this record is injected in the Update.esm and does not exist in this file directly

Heavy Materials		
Name	Keyword	Master
Ancient Nord	WAF_ArmorMaterialDraugr	Update.esm *
Blades	ArmorMaterialBlades	Skyrim.esm
Bonemold Heavy	DLC2ArmorMaterialBonemoldHeavy	Dragonborn.esm
Chitin Heavy	DLC2ArmorMaterialChitinHeavy	Dragonborn.esm
Daedric	ArmorMaterialDaedric	Skyrim.esm
Dawnguard	DLC1ArmorMaterialDawnguard	Dawnguard.esm
Dawnguard Hunter	DLC1ArmorMaterialHunter	Dawnguard.esm
Dragonplate	ArmorMaterialDragonplate	Skyrim.esm
Dwarven	ArmorMaterialDwarven	Skyrim.esm
Ebony	ArmorMaterialEbony	Skyrim.esm
Falmer	ArmorMaterialFalmerHeavyOriginal	Dawnguard.esm
Falmer Hardened	ArmorMaterialFalmerHardened	Dawnguard.esm
Falmer Heavy	kwArmorMaterialFalmerHeavy	Dawnguard.esm
Golden	ccBGSSSE025_ArmorMaterialGolden	ccBGSSSE025-AdvDSGS.esm
Imperial Heavy	ArmorMaterialImperialHeavy	Skyrim.esm
Iron	ArmorMaterialIron	Skyrim.esm
Iron Banded	ArmorMaterialIronBanded	Skyrim.esm
Madness	ccBGSSSE025_ArmorMaterialMadness	ccBGSSSE025-AdvDSGS.esm
Nordic	DLC2ArmorMaterialNordicHeavy	Dragonborn.esm
Orcish	ArmorMaterialOrcish	Skyrim.esm
Stalhrim Heavy	ArmorMaterialStalhrimHeavy	Dragonborn.esm
Steel	ArmorMaterialSteel	Skyrim.esm
Steel Plate	ArmorMaterialSteelPlate	Skyrim.esm
Wolf	WAF_ArmorWolf	Update.esm *

* this record is injected in the Update.esm and does not exist in this file directly

Renamer

Any combination of the options is allowed (including none), however some flags have limited application by themselves and are more effective when combined with others.

→ **i** flag: make search case-insensitive

```
{ "find": "Sword", "replace": "Broadsword", "options": "i" }  
/* will find and replace not only 'Sword', but also variants  
   with other cases - 'sword', 'SWORD', etc. */
```

→ **g** flag: search and replace all instances of the substring

```
{ "find": "Sword", "replace": "Broadsword", "options": "g" }  
/* will find and replace each 'Sword' in the 'Sword of the  
   Sword Singers', and not just the first match */
```

→ **p** flag: allow substring as the part of a word

```
{ "find": "Sword", "replace": "Blade", "options": "p" }  
// will find and replace 'Sword' in 'The Swordsman's Blade'
```

→ **c** flag: retain original case

```
{ "find": "Sword", "replace": "blade", "options": "c" },
// will replace 'Sword' with 'Blade' (with uppercase b)
{ "find": "sword", "replace": "Blade", "options": "c" }
// will replace 'sword' with 'blade' (with lowercase b)
```

→ **n** flag: search for the next rule

```
{ "find": "Woe", "replace": "Grief", "options": "" }
/* will replace 'Woe' in the 'Dagger of Woe' and will not
   look for any more rules */
{ "find": "Blade", "replace": "Dagger", "options": "n" }
/* will replace 'Blade' in the 'Blade of Woe', and will
   search for the next matching rule */
```

Note that ReProcce iterates rules lists from the bottom to the top

→ **o** flag: allow weapons with overridden type (weapon only flag)

```
{ "find": "Sword", "replace": "Broadsword", "options": "o" }
/* will replace 'Sword' in the 'Akaviri Sword' and 'Blades Sword';
   with no 'o' flag these weapons will be ignored, because there
   are typeOverride rules for them (by default) */
```


Localizing ReProccer Reborn

ReProccer Reborn 0.23+ has the localization “framework” that allows you to translate the patcher to your language in a quick and simple manner.

Requires a translated version of Skyrim AE Redone 2.0+

General info

Translations, as well as the rules, are stored in JSON format in the **reproccerReborn/locales** folder, and can be modified with any text editor. By default, the folder has only 1 file:

 **english.jsonc** - this file contains original English strings; keep in mind that English strings are also fallback strings, i.e. if something is wrong with a string in your language, an original English one from this file will be used, so the file should always be present in the **locales**.

Copy and rename it to your game language (french.jsonc, polish.jsonc, etc.)

The locales folder could contain any number of translation files, but only 2 will be loaded simultaneously - English, and the one for your game's language.

The translation file structure is similar to the one in rules.

Language

The first **key** of the translation file is language (the word “english” in the example below).

```
{  
  "english": {  
    "key": "value",  
    "key": "value",  
    "key": "value"  
    ...  
  }
```

This key should match the game’s language (spanish, polish, etc.).

Note, that while technically a translation file could contain strings for several languages, doing so impairs the file’s readability, and is not recommended.

Strings

Each string consists of the **pointer** to the string and the **string** itself.

```
{  
  "english": {  
    "name_enhanced": "Enhanced",  
    "name_recurve": "Recurve",  
    "name_lweight": "Lightweight"  
    ...  
  }
```

Pointers are used in the ReProccer code to get strings from the translation file. **Never modify pointers!** Only translate **strings**.

- For your language string could consist of several words (Longspear -> Długa włócznia).
- For your language some strings could be duplicated (Shortbow/Short Bow -> Короткий лук).

Nouns with gr. gender

If your language has such a category as grammatical gender, and adjectives' form depends on grammatical gender (like in Polish or Russian), you need to take care of correct naming for some ReProccer-generated records (currently only Refined Silver weapons require this).

To achieve this, ReProccer matches a noun's grammatical gender with the adjective of the same gender. The `genderedNouns` key stores nouns by grammatical genders. For instance, in Polish it could look like this:

```
"genderedNouns": {
  "m": ["Młot bojowy"],
  "f": ["Buława"],
  "n": []
}
```

Młot bojowy (Warhammer) in Polish has masculine gender, and will have an adjective with masculine suffix; buława (Mace) has feminine gender and will have an adjective with feminine suffix.

E.g.:

- **Refined** Silver Warhammer -> Wytworny srebrny młot bojowy
- **Refined** Silver Mace -> Wytworna srebrna buława

Note that if your language has no grammatical genders, or if nouns does not affect the form of adjectives, **genderedNouns** should be set to `{} (curly brackets)`:

```
"genderedNouns": {}
```

Gendered adjectives

ReProccer will use an adjective corresponding to the noun's gender, if the noun is found in the records name:

```
"polish": {
  "name_refined": "Wytworny",
  "name_refined_adj": ["Wytworny", "Wytworna", "Wytworne"],
  "desc_refined": "Ta doskonała broń podpala nieumarłych wrogów, zadając dodatkowe obrażenia"
```

In this example, ReProccer will use a proper value from **name_refined_adj** if a noun of masculine, feminine, or neuter gender will be found in the name of the processed record - otherwise, default value (from **name_refined**) will be used.

- The order of genders in the list is masculine, feminine, neuter.
- You don't need to fill adjective lists for languages with no grammatical gender.

Rules translation

Rules should also be translated.

I advise not to remove original English rules from rules files.

While it is safe to do so, leaving original rules intact will help you to close translation gaps - in cases when your game has untranslated strings (or untranslated mods overall), original rules will take care of them.

Note, that some values in rules should never be translated (refer to the relevant item for details):

- [weapon materials](#)
- [weapon material overrides](#)
- [weapon types](#)
- [weapon type overrides](#)
- [armor materials](#)
- [armor material overrides](#)
- [armor faction keywords](#)