

Item Tweaker

Contents

1. Introduction	2
1.1. Compatibility	2
1.2. Installation	3
1.3. Configuration	3
1.4. General usage overview	4
2. Selectors.....	7
2.1. Basic selector structure	7
2.2. Examples of selectors with basic expressions	9
2.3. Selector structure with logical expressions	12
2.4. Examples of selectors with logical expressions	14
3. Nested properties	16
3.1. In "set" and "multiply"	16
3.2. As a "key"	18
4. Manual Overwrite.....	19
5. More useful examples which you can use.....	20
5.1. Selectors.....	20
5.2. Manual Overwrites	27

1. Introduction

This mod allows you to edit item properties by making a query to select a certain range of items and applying changes to items in that selection.

There are two types of query expressions which allow you to filter items – basic expressions and logical expressions. With basic expressions you can set a condition with a property key, comparing operation and an array of values. With logical expressions you can combine several basic expressions and join them with a logical condition “AND” or condition “OR”.

For most use cases you won't need to build intricate, complicated query trees and you can easily achieve whatever you want.

In case you are familiar with the structure of JSON files and logical operations, simply skim over the basics and go to section 5 with examples to get an idea how to use this mod.

If, however, you struggle to understand how to achieve desired changes, this document contains detailed information and examples which should (hopefully) explain to you how to get a certain group of items and apply the changes you want.

1.1. Compatibility

Item tweaker only changes properties of items which you filter out with configured queries. If verbose logging is enabled you will see the old value and the newly applied value. It's recommended to load this mod last, to make sure that its changes to the database aren't overwritten by another mod. If some different mod changes an item which you want to edit, Item tweaker will apply changes (multipliers or overwrite a value) on top of it.

1.2. Installation

Simply extract/copy the mod folder into “/user/mods” directory inside your SP Tarkov folder, so that the folder structure looks as follows:

(D:) > SPT-AKI > SPTARKOV > user > mods > Nightingale-itemtweaker-1.1.0 >

Name	Date modified	Type
config	3/15/2023 1:55 AM	File folder
src	3/15/2023 1:55 AM	File folder
LICENSE	4/20/2021 2:38 AM	File
package.json	3/13/2023 2:43 AM	JSON Source File
Readme.pdf	3/14/2023 6:48 PM	Chrome HTML Do...

1.3. Configuration

The config directory contains three JSON files which allow you to configure the mod – config, dynamic_selectors and manual_overwrite.

File *config.json* has an option to turn on/off verbose logging during mod’s execution, which might be helpful to provide more detailed info to debug what is being done, but might flood the console with lots of lines.

File *dynamic_selector.json* contains selectors. Here you define which items you want to change and how.

File *manual_overwrite.json* contains high priority changes to individual items determined by their name. High priority means that changes in this file will take priority over any changes in selectors if they affect the same item, without overlapping.

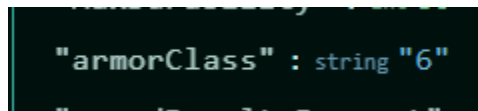
1.4. General usage overview

Using VSCode/VSCodium or some other editor which supports JSON format is recommended to avoid mistakes and make editing easier and more comfortable.

Selectors should be used if you want to more than one item, e.g. weapons that use 7.62x51 round. If you want to change only one item perhaps you should use manual overwrite, because it's way simpler (described in section 4).

When writing property names and values, keep in mind that **letter case matters** and **new value types have to be identical to the old ones**.

A good example is “armorClass” property in armors:



```
"armorClass" : string "6"
```

As you can see it has a value “6” of type string, so you won’t be able to multiply it and if you set a new value it also has to be a string.

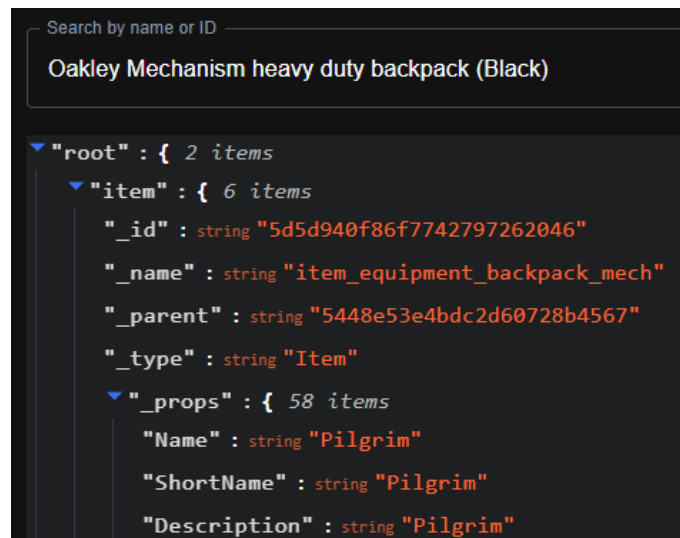
For reference you can use the following:

db.sp-tarkov.com/search – to look for items and see their properties (might have old data).

"Aki_Data\Server\database\templates\items.json" – your actual database file with items, be careful to not edit it, just in case.

hub.sp-tarkov.com/doc/entry/48-recoil-values-and-what-they-do/ – info about weapon recoil values.

Keep in mind that database has inconsistencies and value duplicates, for example the Mechanism backpack has several string properties with value “Pilgrim”:

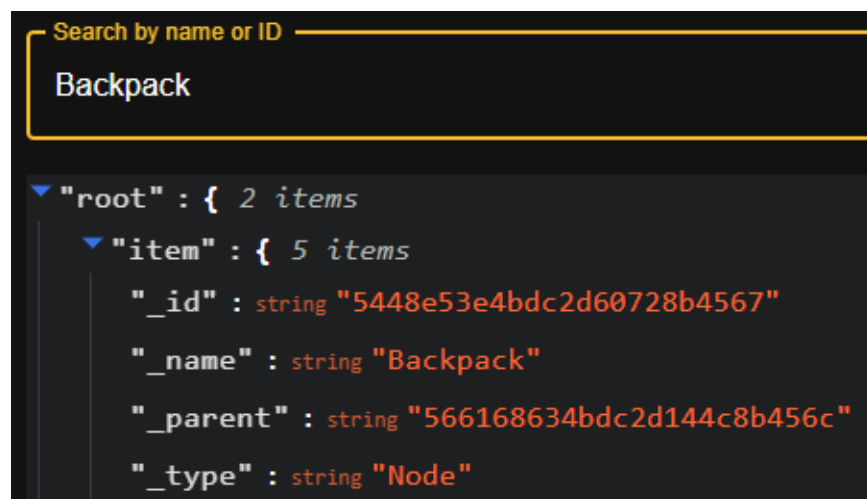


The screenshot shows a search bar with the text "Oakley Mechanism heavy duty backpack (Black)". Below the search bar, a JSON object is displayed. The root object has a key "root" with a value of { 2 items}. The first item in the root array has a key "item" with a value of { 6 items}. The first item in the "item" array has a key "_id" with a value of "5d5d940f86f7742797262046", a key "_name" with a value of "item_equipment_backpack_mech", a key "_parent" with a value of "5448e53e4bdc2d60728b4567", a key "_type" with a value of "Item", and a key "_props" with a value of { 58 items}. The first item in the "_props" array has a key "Name" with a value of "Pilgrim", a key "ShortName" with a value of "Pilgrim", and a key "Description" with a value of "Pilgrim".

```
Search by name or ID
Oakley Mechanism heavy duty backpack (Black)

{"root": { 2 items
  "item": { 6 items
    "_id": string "5d5d940f86f7742797262046"
    "_name": string "item_equipment_backpack_mech"
    "_parent": string "5448e53e4bdc2d60728b4567"
    "_type": string "Item"
    "_props": { 58 items
      "Name": string "Pilgrim"
      "ShortName": string "Pilgrim"
      "Description": string "Pilgrim"
```

One good way to group items by a category is to use their “_parent” id. Here you see that Mechanism has a parent id "5448e53e4bdc2d60728b4567". If you search this id in the resources you’ll see that it’s a node called "Backpack".



The screenshot shows a search bar with the text "Backpack". Below the search bar, a JSON object is displayed. The root object has a key "root" with a value of { 2 items}. The first item in the root array has a key "item" with a value of { 5 items}. The first item in the "item" array has a key "_id" with a value of "5448e53e4bdc2d60728b4567", a key "_name" with a value of "Backpack", a key "_parent" with a value of "566168634bdc2d144c8b456c", and a key "_type" with a value of "Node".

```
Search by name or ID
Backpack

{"root": { 2 items
  "item": { 5 items
    "_id": string "5448e53e4bdc2d60728b4567"
    "_name": string "Backpack"
    "_parent": string "566168634bdc2d144c8b456c"
    "_type": string "Node"
```

A quick way to see if everything works as you wanted it to is to run the mod without verbose logging and confirm that number of changes and items changed make sense considering your configuration.

```

Item Tweaker: Starting...
Initialization...
Applying Selector Tweaks...
"armors_mousePenalty_to_zero" made 103 changes to 103 items
"armors_weight_half" made 91 changes to 91 items
"dmsr_recoil_tweaks" made 7 changes to 7 items
"reduce_all_weapons_camera_recoil" made 125 changes to 125 items
"all_helmets_dont_block_earpiece" made 18 changes to 18 items
"faceshields_dont_block_nvz" made 27 changes to 27 items
"sa58_mdr_762x51_recoil_tweaks" made 4 changes to 2 items
"scar1_and_scarh_tweaks" made 8 changes to 4 items
"9x39_tweaks" made 4 changes to 2 items
"pistol_tweaks" made 46 changes to 23 items
"backpacks_weight_less" made 34 changes to 34 items
"unarmored_rigs_weight_less" made 27 changes to 27 items
"double_all_amm0_stack_size" made 173 changes to 173 items
"socom_mods_tweaks" made 3 changes to 3 items
Applying Manual Overwrite Tweaks...
Manual Overwrite made 2 changes to "weapon_izhmash_svd_s_762x54"
Manual Overwrite made 2 changes to "backpack_Raid_6SH118"
Item Tweaker: Completed

```

Verbose logging outputs much more detailed info about the whole process and outputs warnings and errors which aren't critical to the function of the mod but prevent the application of certain changes. In this example that would be if some items simply didn't have the "mousePenalty" property, not a big deal, the mod will simply skip that property change in the item. Whether these "skips" actually matter depends completely on what you want to achieve.

```

Item Tweaker: Starting...
Initialization...
Applying Selector Tweaks...
Applying "armors_mousePenalty_to_zero"...
Item: hats_UF_PRO - id: 5aa2ba46e5b5b000137b758d
| mousePenalty: Default or identical value used (0). No changes applied.
Item: item_equipment_helmet_maska_1sh - id: 5c091a4e0db834001d5addc8
| mousePenalty: Successfully applied value 0 (was -3)
Item: item_equipment_head_beret_blk - id: 5f60e6403b85f6263c14558c
| mousePenalty: Default or identical value used (0). No changes applied.
Item: item_equipment_cap_r2 - id: 5f99418230835532b445e954
| mousePenalty: Default or identical value used (0). No changes applied.
Item: jack_o_lantern - id: 59ef13ca86f77445fd0e2483
| mousePenalty: Successfully applied value 0 (was -2)
Item: helmet_ratnik_6B47 - id: 5a7c4850e899ef00150be885
| mousePenalty: Successfully applied value 0 (was -2)
Item: item_equipment_head_bandana - id: 5b43271c5acfc432ff4dce65
| mousePenalty: Default or identical value used (0). No changes applied.
Item: cap_mhs - id: 5aa2b89be5b5b0001569311f
| mousePenalty: Default or identical value used (0). No changes applied.
Item: helmet_wilcox_skull_lock - id: 5a16bb52fcdcb001a3b00dc
| mousePenalty: Default or identical value used (0). No changes applied.
Item: ushanka - id: 59e7708286f7742cbd762753
| mousePenalty: Default or identical value used (0). No changes applied.
Item: helmet_crew - id: 5df8a58286f77412631087ed
| mousePenalty: Default or identical value used (0). No changes applied.
Item: helmet_armasight_nvz_goggles_mask - id: 5c066ef40db834001966a595
| mousePenalty: Default or identical value used (0). No changes applied.

```

2. Selectors

2.1. Basic selector structure

When defining selectors you need to follow a proper structure. Here is the structure of a selector with a simple query.

```
"any_selector_name": {
  "query": {
    "key": "item_property_key",
    "operation": "greater_than"/"less_than"/"equals"/"starts_with"/"contains"/"ends_with",
    "values": [
      "value1",
      "value2",
      ...other values
    ],
    "negation": true/false,
    "strict": true/false
  },
  "multiply": {
    "propertyKey1": 69, (any number)
    ...other changes
  }
  "set": {
    "propertyKey1": 69, (any value of the same type as target property)
    ...other changes
  }
}
```

Lets go over each line in detail:

- **"any_selector_name"** – in this key you simply define **selector's name** which you desire.
- **"query"** – define **a query with conditions** to select items. In this structure example a basic expression with one condition and a set of values is provided, later a more complex query structure will be shown.
- **"key"** – define a property key (**name of the property**) which should be compared.

- **"operation"** – defines the comparison **condition to filter items**. Can have the following values: *"greater_than"*, *"less_than"*, *"equals"*, *"starts_with"*, *"contains"*, *"ends_with"*. Can be skipped and if it is – by default uses *"equals"* condition.
- **"values"** – has to be **an array of values** which are **used with "operation"**.
- **"negation"** – simply a logical **"not" operator**. Can be *true* or *false*. Used to invert the condition. Can be skipped, by default has value *false*.
- **"strict"** – determines whether condition should be true for every (logical *"and"*) or at least one (logical *"or"*) of the provided values. Can be *true (every)* or *false (at least one)*. E.g. all . Can be skipped, by default has value *false*.
- **"multiply"** – contains **property names and their multipliers**. Can only be applied to item properties of type *"number"*. Can be skipped if you don't want to multiply anything.
- **"set"** – contains **property names and values you want to set**. New value type has to correspond to target property type. Can be skipped if you don't want to set anything.

Note: Values in the *"set"* object are applied after the multipliers, so if they affect the same properties, multiplier changes will be overwritten with set values.

Note: While unlikely for most use cases, selectors might intersect if they edit the identical properties in identical items. Changes will still be applied but you will be informed to expect overlapping changes. Such conflicts can be resolved by manually applying changes to conflicting items individually in the *manual_overwrite.json*. Later I might introduce a priority system for selectors.

2.2. Examples of selectors with basic expressions

A selector which tweaks pistols. Looks for items where “weapClass” property has value “pistol”. Multipliers are set to 0.15, therefore vertical and horizontal recoil are reduced by 85%.

```
"pistol_tweaks": {
  "query": {
    "key": "weapClass",
    "values": [
      "pistol"
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.15,
    "RecoilForceBack": 0.15
  }
}
```

A selector, which removes mouse sensitivity, changes from any armor or clothing. Looks for items where “ArmorType” property has value “Light” or “Heavy”, or “None”. Simply sets the “mousePenalty” to 0.

```
"armors_mousePenalty_to_zero": {
  "query": {
    "key": "ArmorType",
    "values": [
      "Light",
      "Heavy",
      "None"
    ]
  },
  "set": {
    "mousePenalty": 0
  }
}
```

Note: Compared to filtering by “_parent”, filtering by all possible values of “ArmorType” is a good way to capture all items, which might have “mousePenalty”,

because this property can be present in any wearable item (helmets, armors, faceshields, aventails, ear armor, helmet plates) have different parents.

A selector which doubles ammo stack sizes. Looks for items with ammo parent id. Multiplies the property “StackMaxSize” by 2. If you play around you can also make other items stackable, by default the ones which don’t stack have a “StackMaxSize” of 1.

```
"double_all_ammo_stack_size": {
  "query": {
    "key": "_parent",
    "values": [
      "5485a8684bdc2da71d8b4567"
    ]
  },
  "multiply": {
    "StackMaxSize": 2
  }
}
```

A selector which makes everything examined by default. A good way to select all items is to simply filter by “_type” property and look for value “Item”.

```
"make_everything_examined": {
  "query": {
    "key": "_type",
    "values": [
      "Item"
    ]
  },
  "set": {
    "ExaminedByDefault": true
  }
}
```

A selector which sort of makes sense. Looks for Ushanka’s id. Gives it appropriate protection qualities so that scavs have a real basis to tank headshots.

```
"so_thats_why_they_tank_headshots": {
  "query": {
    "key": "_id",
```

```


    "values": [
        "59e7708286f7742cbd762753"
    ],
    },
    "set": {
        "Durability": 69,
        "MaxDurability": 69,
        "armorClass": "6",
        "armorZone": ["Head"],
        "headSegments": ["Top", "Nape", "Ears"],
        "RicochetParams": {
            "x": 0.9,
            "y": 0.4,
            "z": 65
        },
        },
        "ArmorMaterial": "UHMWPE",
        "ArmorType": "Heavy"
    }
}

```

Ushanka ear flap hat

Gear > Headgear

0.100Kg



ADD TO W-LIST

FILTER BY ITEM

LINKED SEARCH

REQUIRED SEARCH

ARMOR POINTS

69

TRADER:

3108

RICOCHET CHANCE

High

ARMOR CLASS

6

MATERIAL

Ultra high molecular weight polyethylene

ARMOR TYPE

Heavy

ARMOR AREAS

The Ushanka ear flap hat is a classic hat worn by Russian people, or that is at least what most people from the west seem to believe.

Ushanka ear flap hat

Total 9 X 30/30

Gear > Headgear

Ushanka ear flap hat

Total 4 X 29/37

Gear > Headgear

Ushanka ear flap hat

Total 9 X 44/54

Gear > Headgear

Ushanka ear flap hat

Total 3 X 45/65

Gear > Headgear

Ushanka ear flap hat

Total 9 X 60/65

Gear > Headgear

Ushanka ear flap hat

Total 5 X 49/64

Gear > Headgear

Ushanka ear flap hat

Total 2 X 69/69

Gear > Headgear

Top

Nape

Ears

Ushanka ear flap hat

Total 5 X 69/69

Gear > Headgear

Ushanka ear flap hat

Total 9 X 69/69

Gear > Headgear

2.3. Selector structure with logical expressions

If you want to filter items with more than one condition you can use a logical expression query to combine them. Make sure the structure is as follows:

```
"scarl_and_scarh_tweaks": {
  "query": {
    "condition": "and"/"or",
    "expressions": [
      {
        "key": "item_property_key",
        "operation": "greater_than"/"less_than"/"equals"/"starts_with"/"contains"...
        "values": [
          "value1",
          "value2",
          ...other values
        ],
        "negation": true/false,
        "strict": true/false
      },
      ...other expressions
    ]
    "negation": true/false,
  },
  "multiply": {
    "propertyKey1": 69, (any number)
    ...other changes
  }
  "set": {
    "propertyKey1": 69, (any value of the same type as target property)
    ...other changes
  }
}
```

There are three properties in a logical expression:

- **"condition"** determines whether all (and) or at least one (or) expression has to be true.
- **"expressions"** property can contain multiple other expressions. **Can contain both *basic expressions* and *logical expressions*.**

- **"negation"** property inverts the result, same as in the basic query. Can be skipped, by default has value *false*.

Query evaluation is recursive, so you can build a query tree of needed depth and make your query as precise as you need. For most use cases one or several basic expressions suffice but that completely depends on what you want to do.

2.4. Examples of selectors with logical expressions

A selector which applies tweaks to SCAR-L and SCAR-H. Looks for items which have a weapon parent id (so we know they are weapons) AND their “_name” contains string “weapon_fn_mk”. This way we affect 4 items – both FDE and Black versions of Mk16 (SCAR-L) and Mk17 (SCAR-H).

If you, for example, wanted to affect only L or H version, you’d just have to check if “_name” contains “weapon_fn_mk16” or “weapon_fn_mk17”.

Note: In the same way you can tweak other item groups. For example Zenit grips. Make sure that items has grip parent id and the “_name” contains “zenit”.

```
"scarl_and_scarh_tweaks": {
  "query": {
    "condition": "and",
    "expressions": [
      {
        "key": "_parent",
        "values": [
          "5447b5f14bdc2d61278b4567"
        ]
      },
      {
        "key": "_name",
        "operation": "contains",
        "values": [
          "weapon_fn_mk"
        ]
      }
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.65,
    "RecoilForceBack": 0.65
  }
},
```

Selector which tweaks unarmored rigs. A pretty simple tweak that affects rigs which offer no protection. Look for items with tactical rig parent id AND “ArmorType” of “None”. Multiplier simply halves the weight.

```
"unarmored_rigs_weight_less": {
  "query": {
    "condition": "and",
    "expressions": [
      {
        "key": "_parent",
        "values": [
          "5448e5284bdc2dcb718b4567"
        ]
      },
      {
        "key": "ArmorType",
        "values": [
          "None"
        ]
      }
    ]
  },
  "multiply": {
    "Weight": 0.5
  }
},
```

3. Nested properties

3.1. In "set" and "multiply"

If you want access a property that is nested inside an array or an object simply specify the path to it while separating each object and property with dots.

For example, we want to change the 6Sh118 raid backpack size. To do it we have to reach a “cellsV” and “cellsH” property inside one of the objects in the “Grids” array.

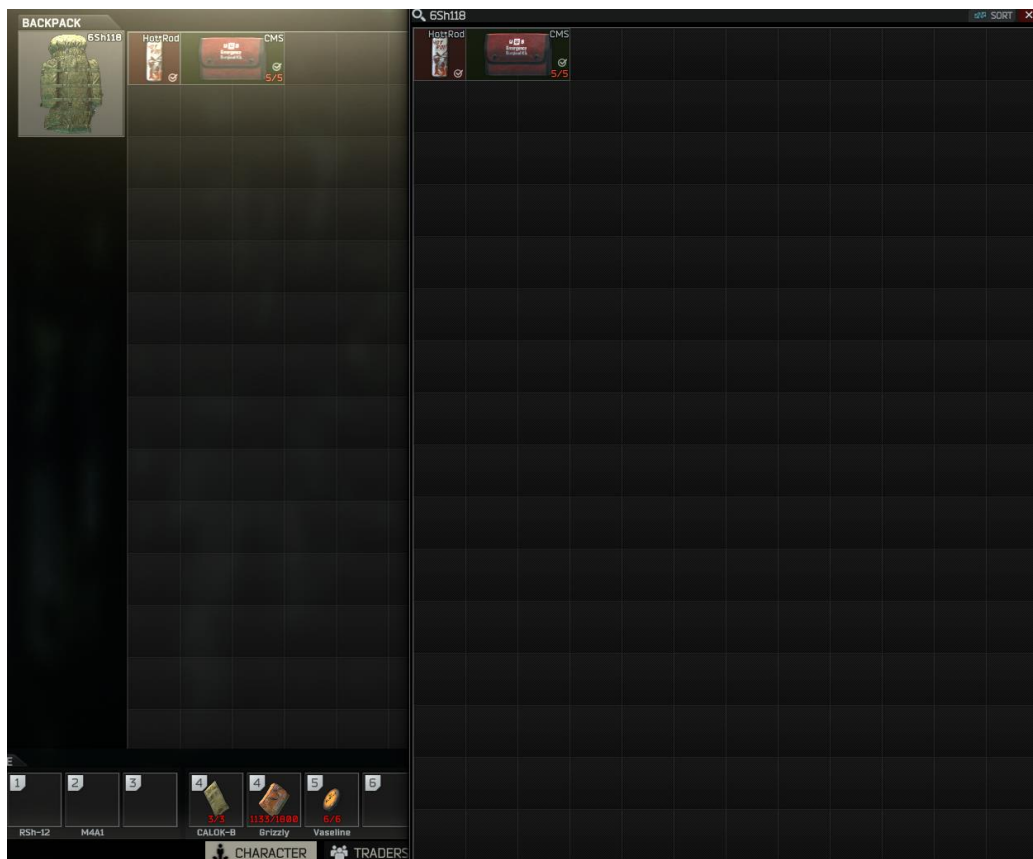
```
▼ "Grids" : [ 1 item
  ▼ 0 : { 5 items
    "_name" : string "main"
    "_id" : string "5df8a4d786f77412672a1e3d"
    "_parent" : string "5df8a4d786f77412672a1e3b"
    ▼ "_props" : { 7 items
      ► "filters" : [...] 1 item
      "cellsH" : int 6
      "cellsV" : int 8
      "minCount" : int 0
      "maxCount" : int 0
      "maxWeight" : int 0
      "isSortingTable" : bool false
    }
    "_proto" : string "55d329c24bdc2d892f8b4567"
  }
]
```


Here you can see that “Grids” contains 1 objects, that’s due to this backpack’s inventory having only one whole container section. Each grid object has “_props” which contains needed properties to change the container size.

A selector which doubles the size of this backpack would look as follows:

```
"double_size_raid_backpack": {
  "query": {
    "key": "_id",
    "values": [
      "5df8a4d786f77412672a1e3b"
    ]
  },
  "multiply": {
    "Grids.0._props.cellsV": 2,
    "Grids.0._props.cellsH": 2
  }
}
```

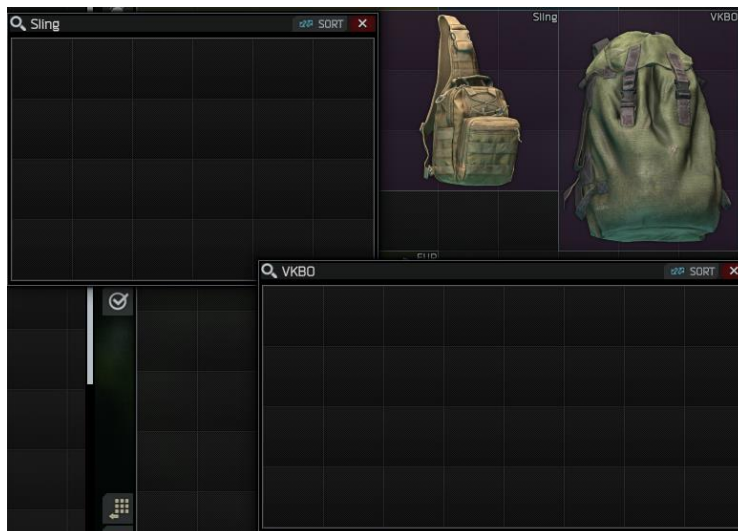
And here is the result:



3.2. As a "key"

The same principle can be applied when filtering items. For example we want to double the size of backpacks where "Grids.0._props.cellsV" is less than 3.

```
"make_some_backpacks_bigger": {
  "query": {
    "condition": "and",
    "expressions": [
      {
        "key": "_parent",
        "values": [
          "5448e53e4bdc2d60728b4567"
        ]
      },
      {
        "key": "Grids.0._props.cellsV",
        "operation": "less_than",
        "values": [
          3
        ]
      }
    ]
  },
  "multiply": {
    "Grids.0._props.cellsV": 2,
    "Grids.0._props.cellsH": 2
  }
}
```



4. Manual Overwrite

All individual item changes in *manual_overwrite.json* take priority over selectors. This file is used to resolve conflicts between selectors or just find one individual item by it's "**_name**" property and apply changes to it.

Basically it's just like a selector but doesn't require a query, only a "_name" as it's key and then your "set" and/or "multiply" changes inside it.

Example of interaction with selectors: selector multiplies the weight of all backpacks by 2, but manual overwrite contains changes to "backpack_sso_attack2" where it sets the multiplier to 4. As a result – the weight for "backpack_sso_attack2" will be multiplied by 4, but for all other backpacks by 2.

For the most part if you want to change only one item you can use manual overwrite. The following JSON block in manual overwrite doubles the 6Sh118 backpack container size just like the selector in section 3.1 did.

```
"backpack_Raid_6SH118": {  
  "multiply": {  
    "Grids.0._props.cellsV": 2,  
    "Grids.0._props.cellsH": 2  
  }  
}
```

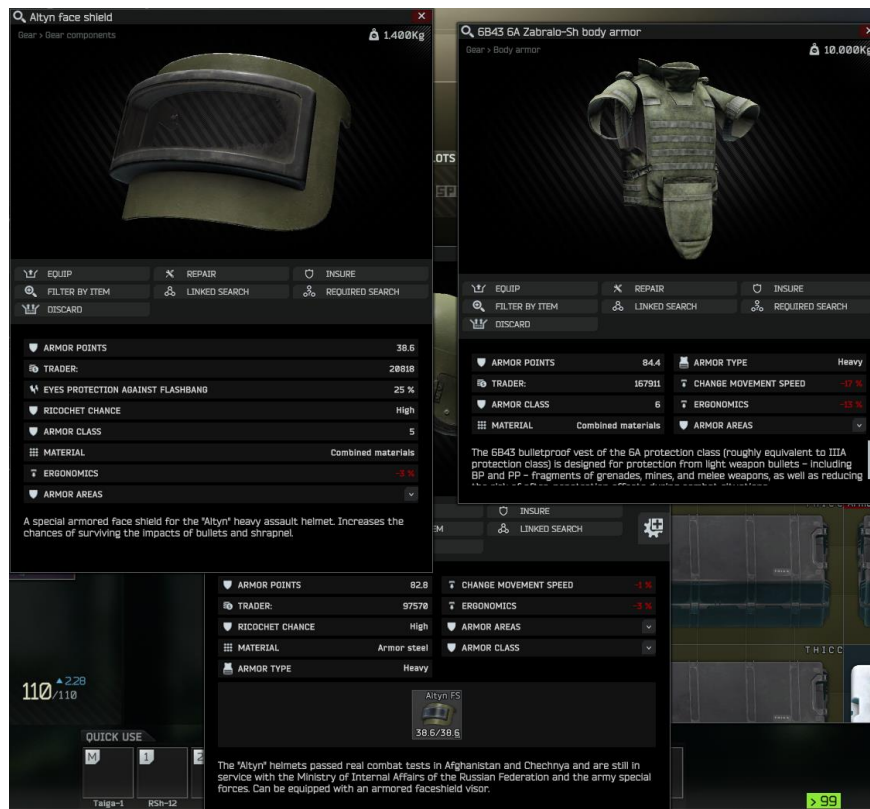
5. More useful examples which you can use

5.1. Selectors

You can also find all these examples in "`\\config\\selector_examples.json`".

Remove turn speed penalty from armors and clothing.

```
"armors_mousePenalty_to_zero": {
  "query": {
    "key": "ArmorType",
    "values": [
      "Light",
      "Heavy",
      "None"
    ]
  },
  "set": {
    "mousePenalty": 0
  }
}
```



Tweak the 9x39 caliber guns (VSS and AS Val).

```
"9x39_tweaks": {
  "query": {
    "key": "ammoCaliber",
    "values": [
      "Caliber9x39"
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.75,
    "RecoilForceBack": 0.75
  }
}
```

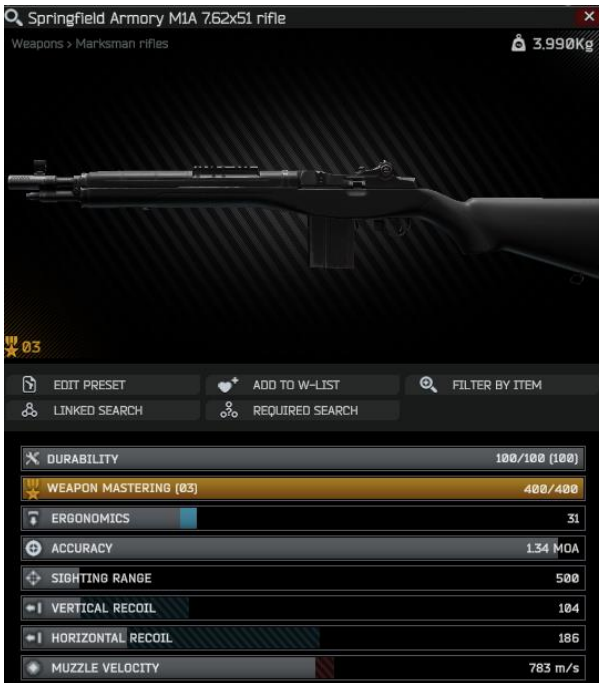
Tweak the SOCOM 16 mods for M1A. Lets say you are satisfied with M1A's stats in a meta build, but you wish to make the SOCOM build more viable. Perhaps a good way to filter items is to check if "_name" contains "socom_16". After all if items don't have the "Recoil" property they'll be ignored.

```
"_name" : string "muzzle_m1a_springfield_armory_socom_16_762x51"
```

```
"_name" : string "stock_m1a_springfield_armory_socom_16"
```

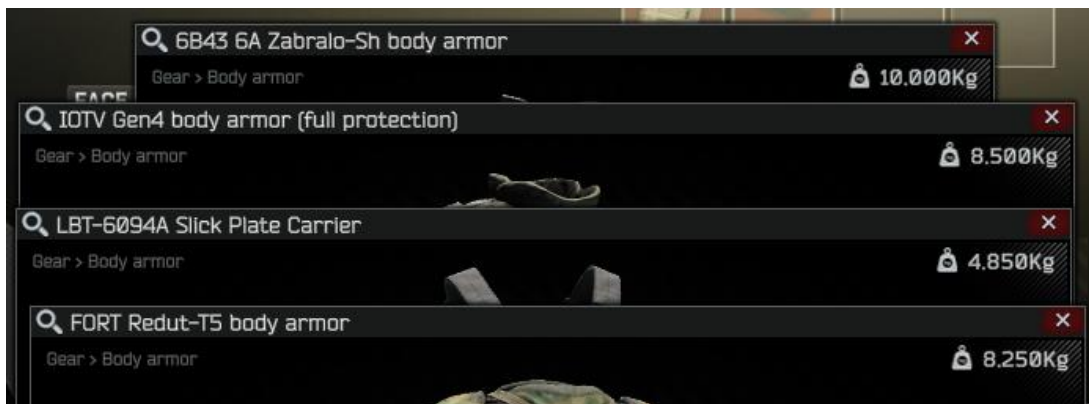
```
"_name" : string "muzzle_m1a_smith_enterprise_socom_16_threaded_gas_lock_762x51"
```

```
"socom_mods_tweaks": {
  "query": {
    "key": "_name",
    "operation": "contains",
    "values": [
      "socom_16"
    ]
  },
  "multiply": {
    "Recoil": 1.5
  }
}
```



Armors weight half.

```
"armors_weight_half": {
  "query": {
    "key": "ArmorType",
    "values": [
      "Light",
      "Heavy"
    ]
  },
  "multiply": {
    "Weight": 0.5
  }
}
```



Change DMRs "RecoilDispersion".

The typo is intended. The recoil stat is invisible, so no screenshots.

```
"dmrs_recoil_tweaks": {
  "query": {
    "key": "weapClass",
    "values": [
      "marksmanRifle"
    ]
  },
  "multiply": {
    "RecoilDispersion": 0.68
  }
}
```

Reduce all weapons camera recoil.

Can't demonstrate this with screenshots.

```
"reduce_all_weapons_camera_recoil": {
  "query": {
    "key": "weapUseType",
    "values": [
      "primary",
      "secondary"
    ]
  },
  "multiply": {
    "CameraRecoil": 0.75
  }
}
```

Allow earpiece with all helmets.

They might clip through, obviously.

```
"all_helmets_dont_block_earpiece": {
  "query": {
    "key": "_parent",
    "values": [
      "5a341c4086f77401f2541505"
    ]
  },
}
```

```

"set": {
  "BlocksEarpiece": false
}

```



Armored Equipment(SLAAPs, Faceshields etc.) and NVGs block nothing.

Basically everything mentioned is compatible with each other. Might clip through, obviously. To turn NVGs on or off with a faceshield on your helmet use context menu (right click on NVGs), because your keybind will only affect the faceshield.

```

"armored_equipment_and_nvgs_block_nothing": {
  "query": {
    "key": "_parent",
    "values": [
      "57bef4c42459772e8d35a53b",
      "5a2c3a9486f774688b05e574"
    ]
  },
  "set": {
    "ConflictingItems": []
  }
}

```



Tweak recoil for SA-58 and MDR 7.62x51.

```
"sa58_mdr_762x51_recoil_tweaks": {
  "query": {
    "key": "_name",
    "values": [
      "weapon_dsa_sa58_762x51",
      "weapon_dt_mdr_762x51"
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.55,
    "RecoilForceBack": 0.55
  }
}
```

Tweak recoil for SCARs (L and H, both FDE and Black).

```
"scarl_and_scarh_tweaks": {
  "query": {
    "condition": "and",
    "expressions": [
      {
        "key": "_parent",
        "values": [
          "5447b5f14bdc2d61278b4567"
        ]
      },
      {
        "key": "_name",
        "operation": "contains",
        "values": [
          "weapon_fn_mk"
        ]
      }
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.65,
    "RecoilForceBack": 0.65
  }
}
```

Tweak pistols recoil.

```
"pistol_tweaks": {
  "query": {
    "key": "weapClass",
    "values": [
      "pistol"
    ]
  },
  "multiply": {
    "RecoilForceUp": 0.15,
    "RecoilForceBack": 0.15
  }
}
```

Reduce backpacks weight by 75%.

```
"backpacks_weight_less": {
  "query": {
    "key": "_parent",
    "values": [
      "5448e53e4bdc2d60728b4567"
    ]
  },
  "multiply": {
    "Weight": 0.25
  }
}
```

Unarmored rigs weight 50% less.

```
"unarmored_rigs_weight_less": {
  "query": {
    "condition": "and",
    "expressions": [
      {
        "key": "_parent",
        "values": [
          "5448e5284bdc2dcb718b4567"
        ]
      },
      {
        "key": "ArmorType",
        "values": [
          "None"
        ]
      }
    ]
  }
}
```

```

    ]
  },
  "multiply": {
    "Weight": 0.5
  }
}

```

Double all ammo stack sizes.

```

"double_all_ammo_stack_size": {
  "query": {
    "key": "_parent",
    "values": [
      "5485a8684bdc2da71d8b4567"
    ]
  },
  "multiply": {
    "StackSize": 2
  }
}

```



5.2. Manual Overwrites

Just a few, maybe already mentioned examples.

```

"weapon_izhmash_svd_s_762x54": {
  "multiply": {
    "RecoilForceUp": 0.55,
    "RecoilForceBack": 0.55
  }
},
"backpack_Raid_6SH118": {
  "multiply": {
    "Grids.0._props.cellsV": 2,
    "Grids.0._props.cellsH": 2
  }
}

```