

1. Modding Documentation .....	2
1.1 Modding Requiem .....	2
1.1.1 REQ-Tags .....	3
1.1.2 Game Mechanics Perks .....	3
1.1.3 Open Encounter Zones .....	4
1.1.4 Compact Leveled Lists .....	4
1.1.5 ActorVariations .....	5
1.1.6 Custom Races .....	5
1.1.7 Tempered Items .....	6
1.1.8 Automated keyword distribution via the Reqtificator .....	7
1.1.9 Follower Integration .....	12
1.2 Reference Documentation .....	12
1.2.1 Reqtificator Configuration .....	13
1.2.2 Keyword Reference .....	15
1.3 Hands-On Tutorials .....	17
1.3.1 Suppress the too many mods warning from the Reqtificator .....	17
1.4 Modding Resources .....	17
1.4.1 Delta Patches for Requiem releases .....	17
1.4.2 Artwork Resources for third-party authors .....	18
1.4.3 EditorID conventions .....	19

# Modding Documentation

If you intend to publish your Requiem-based works, please read [permissions in the mod's Nexus description](#). We're not putting many constraints on derivative-works, but the few rules posted there make our life a lot easier.

## General Modding Resources

- [Artwork Resources for third-party authors](#) — This page contains a collection of Requiem-based artworks you can use to decorate your Requiem patches, Let's Plays and other Requiem-related content.
- [Delta Patches for Requiem releases](#) — Here you can find Delta patches for each Requiem release. These files contain all the records that have been added, removed or changed compared to the previous Requiem release.
- [EditorID conventions](#) — Many of the EditorIDs in Requiem follow a convention to improve the navigation through the plugin. The parts of the convention we consider mature and unlikely to change are documented here.

## Reference Documentation

- [Keyword Reference](#) — A list of all the keywords added by Requiem that are of special interest for modders
- [Reqtificator Configuration](#) — Several aspects of the Reqtificator's behavior can be adjusted by third-party mods via external configuration. This configuration allows you to change how the Reqtificator treats the player record and also allows you to suppress any non-critical load order check.

## Hands-On Tutorials

- [Suppress the too many mods warning from the Reqtificator](#) — This hands-on tutorial shows how to configure the Reqtificator's features at the example of disabling a particular setup check during the installation process.

## Modding Requiem

A collection of guides for Requiem-specific modding features.

- [ActorVariations](#)  
ActorVariations allow you to increase the visual diversity of your NPCs. You define a set of "skill template" actors which determine the gameplay behavior and a set of "look template" actors which define the looks of the actor. The Reqtificator then creates a LeveledChar with all possible combinations of these templates for you to use.
- [Automated keyword distribution via the Reqtificator](#) Keyword checks play a key role in Skyrim's condition system. Unfortunately, it doesn't usually allow fine grained checks like "does this actor have a heavy armor cuirass?". The Reqtificator allows you to distribute composite keywords to build such powerful conditions.
- [Compact Leveled Lists](#)  
Compact Leveled Lists allow you to define leveled items and characters with fine-tuned spawn chances for different entries without tedious boilerplate. This feature only works for leveled lists that are supposed to spawn at most one item.
- [Custom Races](#)

## Modding Guides for Requiem-specific Features

- [ActorVariations](#) — ActorVariations allow you to increase the visual diversity of your NPCs. You define a set of "skill template" actors which determine the gameplay behavior and a set of "look template" actors which define the looks of the actor. The Reqtificator then creates a LeveledChar with all possible combinations of these templates for you to use.
- [Automated keyword distribution via the Reqtificator](#) — Keyword checks play a key role in Skyrim's condition system. Unfortunately, it doesn't usually allow fine grained checks like "does this actor have a heavy armor cuirass?". The Reqtificator allows you to distribute composite keywords to build such powerful conditions.
- [Compact Leveled Lists](#) — Compact Leveled Lists allow you to define leveled items and characters with fine-tuned spawn chances for different entries without tedious boilerplate. This feature only works for leveled lists that are supposed to spawn at most one item.
- [Custom Races](#) — Custom Races require a few changes to be compliant with Requiem.
- [Follower Integration](#) — Followers from third-party mods which do not use the Vanilla Skyrim follower system need some handling to become available in some of Requiem's follower-specific features.
- [Game Mechanics Perks](#) — The Reqtificator can distribute perks to all actors. Use this feature to introduce game mechanics that apply to all actors in the game, but require perks for their implementation.
- [Open Encounter Zones](#) — Encounter Zones are open by default in Requiem and enemies will follow you outside of their dungeons. If you want to restrict certain encounters to remain in their cells, you can flag an encounter zone as exception.
- [REQ-Tags](#) — You can add REQ-Tags to the description of your plugin to unlock advanced modding features provided by the Reqtificator.
- [Tempered Items](#) — Requiem allows you to distribute tempered items to NPCs with a fine control over the magnitude of the applied tempering bonuses.

Custom Races require a few changes to be compliant with Requiem.

- [Follower Integration](#)

Followers from third-party mods which do not use the Vanilla Skyrim follower system need some handling to become available in some of Requiem's follower-specific features.

- [Game Mechanics Perks](#)

The Reqticator can distribute perks to all actors. Use this feature to introduce game mechanics that apply to all actors in the game, but require perks for their implementation.

- [Open Encounter Zones](#)

Encounter Zones are open by default in Requiem and enemies will follow you outside of their dungeons. If you want to restrict certain encounters to remain in their cells, you can flag an encounter zone as exception.

- [REQ-Tags](#)

You can add REQ-Tags to the description of your plugin to unlock advanced modding features provided by the Reqticator.

- [Tempered Items](#)

Requiem allows you to distribute tempered items to NPCs with a fine control over the magnitude of the applied tempering bonuses.

## REQ-Tags

You can add REQ-Tags to the description of your plugin to unlock advanced modding features provided by the Reqticator.

The available features are:

- ActorVariations - take a template actor and combine them with multiple visual templates to increase the variation among encounters
- Tempered Lists - create Leveled Lists containing tempered items
- Compact Leveled Lists - simplified management of spawn ratios for Leveled Lists which spawn one item

To make use of these features in your mod, you need to add some metadata for the Reqticator to your plugin's description like this:

```
This patch makes fancy mod X fully compatible with Requiem.
```

```
Built for:
```

```
Requiem 1.8.2
```

```
Fancy Mod X 42.0
```

```
<<REQ:UNROLL; REQ:TEMPER>>
```

The code in the last line is the REQ-Tag declaration. The tag declaration is delimited by the double angled brackets and the tags are separated by semi-colons. You can add more REQ-Tags to switch on the features you want to use:

- REQ:TEMPER - register the mod for the processing of [Tempered Items](#)
- REQ:UNROLL - register the mod for the processing of [Compact Leveled Lists](#)
- REQ:MUTATE - register the mod for the processing of [ActorVariations](#)



The REQ-tags are properties of the mod that **defines** a record. You therefore cannot opt-in records originating from other mods for these features by renaming their EditorIDs in your plugin. It also means that you can safely modify records from REQ-tagged mods in your patches.



You may see some plugins that have a REQ-Tag definition like: `<<REQ:"FANCY"; REQ:UNROLL; REQ:TEMPER>>`. This is an old and deprecated declaration of the REQ-tags. For the time being it will write a warning to the log files and then continue patching. The first tag denoted a special prefix that you needed to use in your editorIDs. This additional safeguard turned out to impose unnecessary constraints without adding real value and has thus been removed.


## Game Mechanics Perks

The Reqticator can distribute perks to all actors. Use this feature to introduce game mechanics that apply to all actors in the game, but require perks for their implementation.

The major advantage of distributing such perks via the Reqticator is that it saves you a lot of tedious and repetitive work and also applies to any additional mods the user has in their load order.

An example how this feature is used by Requiem itself is the armor penetration system implementation from Requiem 3.0.0. The perks you can select in the skill-tree only grant expertise points, but the actual conversion of expertise points into armor penetrating attacks is done by a set of hidden perks that are distributed to every actor via the Reqtificator. Another example is the perk that grants silver weapons additional damage against undead, the corresponding Vanilla feature is restricted to the player only.

All you need to do make use of this feature is adding your perks to the FormList `xx8F57EA <REQ_Skyproc_ReqtificatorPerks>` and the Reqtificator will distribute it to all NPCs who don't inherit their perklist from a template.


 At present (Requiem 3.0.1) the Reqtificator only uses the last imported version of the form list. If multiple Requiem add-ons add perks to this list, some of them won't be distributed. A fix for this is scheduled for Requiem 3.0.2.

Open Encounter Zones

Encounter Zones are open by default in Requiem and enemies will follow you outside of their dungeons. If you want to restrict certain encounters to remain in their cells, you can flag an encounter zone as exception.

If a cell is assigned to an [encounter zone](#), any NPC in it will not move to cells that belong to another encounter zone (or none) unless their zone is flagged as having "open combat boundaries". The Reqtificator sets this flag on all encounter zones to avoid weird situation where a bandit mob stops chasing you just because we passed through the load door of their den.

In some situations however, you may want to keep the closed encounter to avoid other awkward situations like vampires running into the sun. Or you may have some complicated quest setup that may break if the involved actors leave the cell. For such cases the Reqtificator has an exception list `xxA46546 <REQ_List_SkyProc_ClosedEncounterZones>`. Any encounter zone in this list will not be modified.

 The Reqtificator internally merges all definitions of the exception list. You do not need to resolve conflicts between multiple mods making changes to this record yourself.

Compact Levelled Lists

Compact Levelled Lists allow you to define leveled items and characters with fine-tuned spawn chances for different entries without tedious boilerplate. This feature only works for leveled lists that are supposed to spawn at most one item.

Leveled Items and Characters are used in Requiem to randomize loot and encounters. The likeliness of spawns can be controlled by adding a specific entry multiple times to a list. However, if you want to achieve a more fine-grained control over spawn ratios, this can quickly become tedious copy&paste work. And tweaking the numbers later is also a time-consuming and boring task.

Requiem alleviates this feature by introducing *compact leveled lists* which avoid all the boilerplate for leveled lists that spawn exactly one item. (You can still use a non-zero non-chance in leveled items if you want.)

To use this feature you must first enable [REQ-Tags](#) for your mod and add the `REQ:UNROLL` tag. After doing so, you can declare your leveled list with a special EditorID to enable this feature.

A leveled item can be declared as compact list by giving it an EditorID `<Prefix>_CLI_<RecordName>`, where `<Prefix>` should be some common identifier for your mod's records (no underscores) `<RecordName>` is the EditorID you'd normally give to the record. Let us consider an example from Requiem: `xxAD3854 <REQ_CLI_Treasure_GemsCheap>`, a leveled item for spawning random gem treasure.

Level	Count	Reference
1	1	00063B42 <GemRuby>
1	6	00063B45 <GemGarnet>
1	6	00063B46 <GemAmethyst>
1	3	0006851E <GemAmethystFlawless>
1	3	00068521 <GemGarnetFlawless>
1	1	00068522 <gemRubyFlawless>

The Reqtificator will replace each entry in this list with `Count`-many entries of the same item with `Count` set to 1. So after processing this list it will contain 1x1 ruby, 6x1 garnets, 6x1 amethysts, 1x1 flawless ruby, 3x1 flawless garnet and 3x1 flawless amethysts. Or in terms of spawn chances:

- 5% chance to spawn a ruby
- 5% chance to spawn a flawless ruby
- 15% chance to spawn a flawless garnet
- 15% chance to spawn a flawless amethyst
- 30% chance to spawn a garnet
- 30% chance to spawn a amethyst

If you want tweak your spawn ratios later, you can quickly adjust the count values in your plugin instead of having to add/remove entries in the leveled item.

Compact Leveled Characters work exactly the same way, the only difference is that their EditorIDs follow the pattern `<Prefix>_CLChar_<RecordName>`.

## ActorVariations

ActorVariations allow you to increase the visual diversity of your NPCs. You define a set of “skill template” actors which determine the gameplay behavior and a set of “look template” actors which define the looks of the actor. The Reqtificator then creates a LeveledChar with all possible combinations of these templates for you to use.

Actor definitions in Skyrim can reuse a lot of logic by using template actors, especially when combined with a Leveled Character in the inheritance chain. However, the existing functionality is not powerful enough to provide an easy way to separate visual appearances and gameplay behavior to further increase the visual variety of generic actors.


ActorVariations provide a means to solve this problem. Look and Skill templates are defined separately and then merged by the Reqtificator into actors the Skyrim engine can use.

To use this feature you must first enable [REQ-Tags](#) for your mod and add the `REQ:MUTATE` tag. After doing so, you can declare your actor variations as leveled characters with a special EditorID to enable this feature.


A leveled character can be declared as actor variation by giving it an EditorID `<Prefix>_LChar_Variations_<RecordName>`, where `<Prefix>` should be some common identifier for your mod's records (no underscores) and `<RecordName>` is the EditorID you'd normally give to the record. Let us consider an example from Requiem: `xx8A8BEC <REQ_LChar_Variations_Bandit_Trickster_06>`, which defines the boss rank trickster bandits.

Level	Count	Reference
1	4	0684F6B2 <REQ_LChar_LookTemplate_Race_Argonian_Male> (LVLC)
1	1	0684F6B3 <REQ_LChar_LookTemplate_Race_Khajiit_Female> (LVLC)
1	2	068A14DE <REQ_LChar_LookTemplate_Race_Argonian_Female> (LVLC)
1	2	068A14DF <REQ_LChar_LookTemplate_Race_Khajiit_Male> (LVLC)
1	1	068A14E0 <REQ_LChar_LookTemplate_Race_Imperial_Female> (LVLC)
1	2	068A14E1 <REQ_LChar_LookTemplate_Race_Imperial_Male> (LVLC)
1	3	068A14E6 <REQ_LChar_LookTemplate_Race_Breton_Female> (LVLC)
...	...	... [more look templates] ...
1	1	068A8BE7 <REQ_Actor_Bandit_Trickster_06> (NPC_)

Every NPC\_ record found in this list is considered a skill template, while every leveled character is considered as a set of look templates.

 The actors you use as skill and look templates can inherit from other actors, but not from Leveled Characters.

The Reqtificator will create new actors for each possible combination of the templates. From the look template all data is taken that belongs to “Inherit Traits” flag and the skill template contributes all other data. You can use the count values of the look templates and the level values of the skill templates to weight the spawn ratios in the generated list. The outcome of this operation is then a LeveledCharacter that spawns boss-rank trickster bandits, but with many different visual appearance options. You can then use this list as a template actor for the actors you want to spawn in the world, just like Vanilla Skyrim does with its leveled characters with a few hard-coded bandit variants.

 When you look at the actual outcome generated by the Reqtificator, you'll see that there are some additional leveled characters generated. These nested records are only used to avoid some limitations of Skyrim's engine and to optimize the reusability of generated records.

## Custom Races

Custom Races require a few changes to be compliant with Requiem.

To make a playable race ready for Requiem, you will need to add `Requiem.esp` as a master to your custom race plugin (or its patch) and then apply the following changes:

- add the following spells to their racial spell list:
  - `xx609AF0 <REQ_Ability_DisableHealthRegen>` – only for living races, vampire races instead receive a race-specific ability that triples their stamina and magicka regeneration rates and boosts the carry weight
  - `xx7E76F4 <REQ_Ability_Stress_MassPenalty>`

- `xx82CC14 <REQ_Ability_NPCAdjustment_MasseEffect>`
- add the following keywords as appropriate:
  - `xx586728 <REQ_KW_DropsBloodKeyword>` if the race has blood suitable as vampire nutrition
  - `000A82BB <Vampire>` and `000D61D1 <IsBeastRace>` for vampire races (the second keyword enables more feral unarmed attacks)
  - `xx5F367F <REQ_KW_MinorKnockdownImmunity>` for vampire races
  - `xx4CF31E <REQ_KW_StrongStomach>` for races that can eat raw meat and bestial stew
- adjust the base stats, the sum of health, magicka and stamina should be ~300 points, each attribute must be larger than 50 points
- adjust the carry weight, for Requiem it should be in the range of 80-150 points depending on the physical strength of the race
- adjust the base regeneration rates:
  - health regeneration – set to 0.2
  - magicka regeneration – usually in the range of 1.0 to 1.2
  - stamina regeneraiton – usually in the range of 1.5 to 1.7
- adjust unarmed damage, usually between 5-10, clawed races like Khajiit can have higher values (e.g. 15)

That's the basic set of changes you have to make to ensure a race has basic compatibility with Requiem. The trickier part is to rebalance any racial abilities your custom race might have to be in line with Requiem's balance.

## Tempered Items

Requiem allows you to distribute tempered items to NPCs with a fine control over the magnitude of the applied tempering bonuses.

### How tempering works in Requiem

In Requiem each tempering level encompasses a whole range of item health/tempering bonus values. This ensures that tempering bonuses remain useful with Requiem's increased armor ratings and weapon damages while retaining useful descriptive labels. The mapping of tier names to tempering bonuses is shown in the following table:

	Quality Tier Name	Item Health	Damage/Armor rating bonus (x2 for cuirass)
1	Well-Made	100% - 210%	0 - 11
2	High-Grade	220% - 330%	12 - 23
3	First-Rate	340% - 450%	24 - 35
4	Exquisite	460% - 570%	36 - 47
5	Master Work	580% - 690%	48 - 59
6	Legendary	700% - ...	60 - ...

When you want to create your own tempered items in Requiem, you need to specify a few values:

- the quality tier `T` determines the overall quality of the spawned item
- the size factor `S` indicates the item type, the allowed values are:
  - `H (=1)` for daggers and other light weapons
  - `N (=2)` for one-handed weapons and all armor pieces
  - `D (=4)` for two-handed weapons, including bows and crossbows
- the distribution type `D` governs how the spawn probabilities are distributed:
  - `rise` gives a higher probability to high values within the specified item health interval
  - `const` gives equal probability to all values within the specified item health interval
  - `fall` gives a higher probability to low values within the specified item health interval

Based on these values, Requiem will determine minimum and maximum item health values to spawn:

$$\begin{aligned}\text{min item health} &= 100\% + 10\% * 3 * (T - 1) * S \\ \text{max item health} &= 100\% + 10\% * (3 * (T * S) - 1)\end{aligned}$$

These tempering ranges are written as `tier{N}_{S}_{D}`, e.g. `tier2_D_fall`.



Distribution types are scheduled for removal in a future Requiem version to simplify the system. We recommend to simply use the `const` distribution.

### Tempering worn armor

Tempering of armors happens via a small script. You can either attach this script directly to an actor or use a permanent spell with this script tied to an magic effect to achieve the same effect.

The script you need is named `REQ_TemperedEquipment` and it needs the following generic properties set:

- **datastorage:** point to alias `TemperingData` of `REQ_Quest_Playerscripts "A Requiem of better Times" [QUEST:xx2F389E]`

- **Version\_Active:** set to REQ\_Internal\_VersionActive [GLOB:xx2F38A0]
- **Version\_Installed:** set to REQ\_Internal\_VersionInstalled [GLOB:xx2F389F]


In addition, it also needs a map of armor keywords to tempering ranges to know which tempering should be applied to different items. This is done with the remaining script properties:

- **Keywords:** a list of all the keywords that trigger tempering
- **quality:** a list of tempering ranges, they are assigned to the keyword with the same index

index	keywords	quality
0	ArmorMaterialIron	tier2_n_const
1	ArmorMaterialSteel	tier2_n_const
2	ArmorHeavy	tier1_n_const

In this example any worn armor items with the iron or steel material keyword would be tempered with the tier2\_n\_const tempering range, while any other heavy armor item would be tempered using tier1\_n\_const. Any item not matching any of the entries in the map will not be tempered.

If you want to use this script as part of the spell, you need a script archetype effect with constant cast type and self delivery type and a corresponding spell.

 At present, only the tier1\_N and tier2\_N are supported for armor tempering, but for all distributions.

### Tempering other items

To temper any item that is not a worn armor, you can create convenient leveled lists which will be processed by the Reqticator. To use this feature you must first enable [REQ-Tags](#) for your mod and add the REQ:TEMPER tag. After doing so, you can declare your leveled list with a special EditorID like REQ\_LI\_Armor\_ElvenCuirass\_Quality1\_N\_rise to enable this feature:

REQ	_LI_	Armor_ElvenCuirass	_Quality1	_N	_rise
a prefix to identify your mod's records		free-form name to identify the purpose of the leveled list	quality tier T	size factor S	distribution D

Put a single instance of the desired item into the leveled list with count and level set to 1. You can then use this item like any other leveled list in your mod. The Reqticator will take care of transforming the leveled list to spawn the tempered items you requested.

Automated keyword distribution via the Reqticator

### The problem we want to solve

When looking for perk effect conditions or magic effect conditions, you often make checks for the keywords of an actor's gear. As long as you use Vanilla Skyrim keywords in these tests, everything is fine and your work will have a high compatibility with other mods. However, this system also has a serious limitation because you cannot enforce two keyword checks on the same item. You can ask "Does this actor wear an item that has `ArmorHeavy`? And does this actor wear an item (potentially different) that has `ArmorCuirass`?", but not "Does this actor wear an item that has both the `ArmorHeavy` and `ArmorCuirass`?" keywords.

To address this problem, you need to create a single combined keyword, `ArmorCuirassHeavy`, and then assign it to all appropriate items. This of course can be rather time-consuming work and it doesn't support gear from any other mod. We faced this problem ourselves when reworking the arrow resistance in Requiem 2.0. To solve this problem, we added a new functionality to the Reqticator which can add new keywords to armor records based on the keywords these records already have.

The implementation effort for such features is thus limited to writing the distribution rules, and the compatibility with other mods is automatically given by the Reqticator.

### How it works - the small example

Each plugin that has Requiem.esp as a master file can provide a file `KeywordAssignments_<plugin_name>.conf` in the folder `SkyProc Patchers\Requiem\Data`. This file contains the distribution rules for your mod. Those can either be completely new rules, or changes to existing rules from Requiem. These files contain two parts: declarations of the to-be-required keywords, and the actual distribution rules, called "features". A feature groups those rules together that belong together. Requiem for example ships with two features: armor changing in combat, and arrow resistance.

Let's take a look at some excerpts from `KeywordAssignments_Requiem.esp.conf`:

```

; Example of a keyword assignment rule
; This rule adds the keyword 'ArmorCuirassHeavy' to any armor record that has both 'ArmorHeavy' and 'ArmorCuirass' keywords
; The rule is applied to all armor records in the mod
; The rule is applied to all armor records in the mod
; The rule is applied to all armor records in the mod

```

## Distributing the Armor Change in Combat keyword

```
armorParts {
    boots = 06C0ED Skyrim.esm
    cuirass = 06C0EC Skyrim.esm
    gauntlets = 06C0EF Skyrim.esm
    helmet = 06C0EE Skyrim.esm
    shield = 0965B2 Skyrim.esm
}

changeDuringCombat = 94EABF Requiem.esp

feature_armorChangeInCombat {
    keywords_none = [ ${ changeDuringCombat } ]
    keywords_any = [ ${ armorParts.shield } ]
    keywords_assign = [ ${ changeDuringCombat } ]
}
```

This little snippet is all we need to distribute the Requiem keyword that allows you to (un)equip gear marked with it in the midst of combat. Let us at first consider the meaning of this snippet and afterwards discuss the syntax.

First, we **declare keywords** that we want to use in our rules:

```
armorParts {
    boots = 06C0ED Skyrim.esm
    cuirass = 06C0EC Skyrim.esm
    gauntlets = 06C0EF Skyrim.esm
    helmet = 06C0EE Skyrim.esm
    shield = 0965B2 Skyrim.esm
}

changeDuringCombat = 94EABF Requiem.esp
```

We can either define simple *variables* like `changeDuringCombat` or create *objects* like `armorParts { ... }` which contain *variables* to group related keywords. Keywords are specified by their 6-digit formID (no load order index) followed by the plugin name, separated by a space.

Then we define the actual **distribution rule**:

```
feature_armorChangeInCombat {
    keywords_none = [ ${ changeDuringCombat } ]
    keywords_any = [ ${ armorParts.shield } ]
    keywords_assign = [ ${ changeDuringCombat } ]
}
```

The Reqtificator recognizes all *objects* whose names begin with "feature\_" as distribution rules. Distribution rules can contain 4 special *variables*:

- `keywords_none` – items must not have any of the listed keywords to qualify for this rule
- `keywords_any` – items must have at least one of the listed keywords to qualify for this rule
- `keywords_all` – items must all of the listed keywords to qualify for this rule



- `keywords_assign` – the list of keywords to add to items that qualify for this rule

As for the syntax rules, the `[...]` denote *lists* (or *arrays*, makes no difference here) and `${ changeDuringCombat }` and `${ armorParts.shield }` are variable substitutions. After reading the configuration files, these will be substituted by their definitions, i.e. the final declaration would be:

```
feature_armorChangeInCombat {
    keywords_none = [ "94EABF Requiem.esp" ]
    keywords_any = [ "0965B2 Skyrim.esm" ]
    keywords_assign = [ "94EABF Requiem.esp" ]
}
```

This would be hard to read! Using variable substitution thus allows you to use expressive names for your keywords instead of having to use the cryptic formIDs that don't tell you anything about the keyword they represent.

To summarize, this example adds the keyword for the armor change in combat to all shields that do not already have it.

If you changed the declaration like this:

```
feature_armorChangeInCombat {
    keywords_none = [ ${ changeDuringCombat } ]
    keywords_any = [ ${ armorParts.shield } ${ armorParts.helmet } ]
    keywords_assign = [ ${ changeDuringCombat } ]
}
```

...then you would end up with a SkyProc patch that allows you to change any shield or helmet in combat.

## Syntax summary

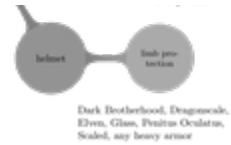
- `objectname {...}` – defines an object, these can be used either for grouping keyword declarations or for defining distribution rules. Objects that are distribution rules must have names beginning with "feature\_".
- `variablename = xxxxxxPlugin.esp` – keyword definition for later use in variable substitutions
- `${variablename}` – variable substitution
- `[...]` – a list of keywords

## Arrow resistance - the complex example

In addition to the simple example shown in the previous section, you can also nest the distribution rules to achieve more powerful declarations. Take a look at our arrow resistance distribution rules below:

### Arrow resistance magic





The first fifty lines are just simple keyword variable declarations for the various materials and armor item types. Our distribution rule `feature_arrowResistance` however is a lot more complex this time. Instead of being a simple object as in the previous example it's actually a tree of nested assignments. If the processed item satisfies the conditions associated with an object, we will add the assigned keywords, and also test if the item satisfies the conditions for any of the nested objects. The condition tree is represented visually in the image on the right.

The top-level node filters out any items that already have arrow-resistance keywords because these are assumed to be edited manually. On the next level we check which armor component we have: boots, gauntlets, helmets or a cuirass? As you can see the helmets, boots and gauntlets must also have a specific materials keyword in addition to the armor component keyword to benefit from arrow resistance.

If we are dealing with a cuirass, we first sub-divide the problem into light and heavy armors. Take a closer look at heavy armors. Here we offer three different levels of protection depending on the material keywords. The two lower ranks explicitly specify the materials to which they apply. On the other hand, the highest rank, which is the default for heavy armors, uses a more sophisticated variable substitution to match all armors that matched neither of the other ranks.

The Reqticator iterates over the entire condition tree and tests all applicable nodes against the *original* keywords of the processed item. Afterwards the to-be-assigned keywords from all matching nodes are added to the record.

Let us consider a few hypothetical examples to clarify this strategy:

- a mixed leather-steel cuirass that is flagged as light armor will receive `arrowResistance.body1` from `feature_arrowResistance.cuirass.light.tier1` (the `arrowResistance.body3` usually added to steel armors doesn't apply because the parent node requires the heavy armor keyword)
- a mixed leather-steel cuirass that is flagged as heavy armor will receive `arrowResistance.body3` from `feature_arrowResistance.cuirass.heavy.tier3` (the `arrowResistance.body1` usually added to leather armors doesn't apply because the parent node requires the light armor keyword)
- a mixed leather-steel cuirass that is flagged as both light and heavy armor will receive `arrowResistance.body1` (from `feature_arrowResistance.cuirass.light.tier1`) and `arrowResistance.body3` (from `feature_arrowResistance.cuirass.heavy.tier3`)

## Changing distribution rules from Requiem, and conflict resolution

You can add your own keyword distribution rules or modify the existing rules from Requiem by providing additional configuration files. The Reqticator will merge the content from `KeywordAssignments_Requiem.esp.conf` with all other configuration files in the same folder belonging to Requiem-dependent plugins.



### Do not modify the config file for Requiem.esp!

You should not modify `KeywordAssignments_Requiem.esp.conf` directly. In this paragraph we explain how you can edit the distribution rules from Requiem by using additional config files. Using a separate configuration file ensures that your mod will not become incompatible with other mods due to file conflicts.

It is important to note that the configuration files are merged before variables are substituted. This means that you can use all the keyword definitions from our master file without having to copy them explicitly.



### If something doesn't work out, see MergedKeywordConfig.txt for help

The Reqticator writes a file named `{{MergedKeywordConfig.txt}}` which contains the final configuration after all merging and variable substitution has been done. This file also contains annotations about the origin of each entry in the configuration.

The Frodo-hack (change rings in combat)

As an example, let us change the armor change in combat feature such that it allows you to also switch rings during combat, but without editing the main configuration file.

### KeywordAssignments\_Frodo Hack.esp.conf

```
armorParts.ring = 10CD09 Skyrim.esm
feature_armorChangeInCombat.keywords_any += ${ armorParts.ring }
```

Now what happens here? In the first line we add the variable `ring` to the object `armorParts` from the main configuration file, since it didn't exist before. (If it did, the previous value would be overwritten by such an assignment.)

In the second line we *append* a new element to the array `keywords_any` in the object `feature_armorChangeInCombat` (note the `+` before the `=`).

After merging the configurations and resolving variables this array now has two entries: FormIDs of the keywords marking shields (which were already there) and rings. In consequence any item marked as a shield or a ring can be (un)equipped during combat.

This dot-notation is quite powerful and allows you to make changes at any level of the original configuration.

Adding Dragonborn materials to the arrow resistance feature

As you saw in the previous example, you can easily add new keyword declarations and extend existing feature definitions. For the sake of the example we'll give them some questionable assignments. We'll add light Chitin armor with the same level of protection as studded armor. We'll also add light Stahlrim armors and declare them to be as powerful as daedric gear, i.e. more powerful than any light armor declared in the main configuration file.

### KeywordAssignments\_Requiem - Dragonborn.esp.conf

```
armorMaterials.chitinlight = 024102 Dragonborn.esm
armorMaterials.stahlrimlight = 024107 Dragonborn.esm

feature_arrowResistance.cuirass.light.tier2.keywords_any += ${
armorMaterials.chitinlight }

feature_arrowResistance.boots.keywords_any += ${ armorMaterials.
stahlrimlight }
feature_arrowResistance.gauntlets.keywords_any += ${ armorMaterials.
stahlrimlight }
feature_arrowResistance.helmet.keywords_any += ${ armorMaterials.
stahlrimlight }
feature_arrowResistance.cuirass.light.tier4 = {
    keywords_assign = [ ${ arrowResistance.body4 } ]
    keywords_any = [ ${ armorMaterials.stahlrimlight } ]
}
feature_arrowResistance.cuirass.light.tier1.keywords_none += ${
feature_arrowResistance.cuirass.light.tier4.keywords_any }
```



First we define the keywords for the new materials, then we add the light Chitin keyword to the `keywords_any` list to ensure that the cuirass gets the desired level of protection, tier2. Tier2 and tier3 nodes are already excluded from tier1 in the main configuration file, so you can see that this will automatically ensure that the tier1 keyword no longer applies.

In the second part we add the light Stahlrim keyword to the material lists for boots, gauntlets and helmet; nothing new happens here.

The cuirass is however an interesting case. Since there's no node in the main configuration file that assigns tier4 arrow protection to light armor, we need to hook in another node which we do by declaring the object `feature_arrowResistance.cuirass.light.tier4`. The last line then updates the exclusion conditions for the default assignment to ensure that our new tier4 light armors (light Stahlrim) don't get the default tier1 protection as well, since they're not already excluded.

The resulting condition tree after merging this configuration file is shown on the right.

## Technical Details and further reading

This implementation is based on the HOCON format (Human-Optimized Configuration Object Notation). If you're curious, you can read more about it [here \(the library we're using\)](#) and [here \(HOCON-specification\)](#).

## Follower Integration

Followers from third-party mods which do not use the Vanilla Skyrim follower system need some handling to become available in some of Requiem's follower-specific features.

Requiem has a centralized script system to handle followers from third-party mods. After registering them once, they can be used in more advanced features like e.g. the lockpicking system.

A NPC must satisfy two conditions to be considered as a follower by Requiem:

- be in a known Reference Alias
- be in the faction `PlayerFollowerFaction` [`FACT:00084D1B`]

Requiem by default only checks the Vanilla follower contained in the alias `Follower` from `DialogueFollower` [`QUST:000750BA`], but you can register additional aliases to be checked.

Requiem supports up to 30 aliases to check and can detect up to four current followers.

## Registering a Follower Alias

To register an Alias for searching followers, you only need to make a simple script call in your mod's setup.

Requiem has a quest `REQ_Quest_FollowerControl` [`QUST:xx82F3AF`] which contains the script `REQ_FollowerRegistration`.

To register your alias, just call the function `RegisterFollowerAlias(ReferenceAlias toRegister)` on the script running on this quest. The function returns `True` if the registration was successful, otherwise it returns `False` and writes the reason to the papyrus log files.

The registration is permanent, i.e. you only need to run this command after the initial setup of your mod.



If you try to register your follower alias directly after a new game has been started, the function call may fail because the management script is not ready yet. In such scenarios you should retry the registration after waiting for some time.

## Unregistering a Follower Alias

To support proper upgrading and deinstallation, the aforementioned system also has an unregister function. To remove your alias from the list of aliases to search, simply call `UnregisterFollowerAlias(ReferenceAlias toUnregister)`. Just as with the registration function, this one returns `True` if the unregistration was successful, otherwise it returns `False` and writes the reason to the papyrus log files.

## Reference Documentation



We also have hands-on tutorials available: [Hands-On Tutorials](#)

- [Keyword Reference](#)


A list of all the keywords added by Requiem that are of special interest for modders

- [Reqtificator Configuration](#)

Several aspects of the Reqtificator's behavior can be adjusted by third-party mods via external configuration. This configuration allows you to change how the Reqtificator treats the player record and also allows you to suppress any non-critical load order check.

## Reqtificator Configuration

Several aspects of the Reqtificator's behavior can be adjusted by third-party mods via external configuration. This configuration allows you to change how the Reqtificator treats the player record and also allows you to suppress any non-critical load order check.

 Also check these hands-on tutorial showing the feature described here in action:

- [Suppress the too many mods warning from the Reqtificator](#)

- [Configuration Management and Conflict Resolution](#)

- [Configuration Storage Location](#)
- [Configuration File Syntax](#)
- [Configuration Conflict Resolution](#)

- [Configurable Features](#)


- [Handling the player record](#)
- [Setup-related Warnings](#)
- [Thresholds for Armor-Rating Adjustments](#)

## Configuration Management and Conflict Resolution

To change the Reqtificator's configuration you need to provide your own configuration file to overwrite the parts of Requiem's base configuration you're interested in.

## Configuration Storage Location

Configuration files are located in `SkyProc Patches\Requiem\Config\<your plugin name without suffix>`. At present the only configuration file is the `Reqtificator.conf`. Requiem's base configuration is therefore `SkyProc Patches\Requiem\Config\Requiem\Reqtificator.conf` and the configuration changed accompanying a plugin `Epic Monsters.esp` would be `SkyProc Patches\Requiem\Config\Epic Monsters\Reqtificator.conf`.

 Do not modify the config file from Requiem itself. This will cause compatibility issues with other mods and future Requiem versions.

## Configuration File Syntax

Requiem's config files use the [HOCON syntax](#), a more human-readable superset of JSON and looks like this:

```
playerRecord {
  healthOffset = 0
  magickaOffset = 0
  staminaOffset = 0
  spellsToRemove = [
    "012FCD Skyrim.esm",
    "012FCC Skyrim.esm"
  ]
}
ignoredWarnings {
  bashedPatchUsed = false
  bugFixesSksePluginMissing = false
  cobbFixesSksePluginMissing = false
  crashFixesSksePluginMissing = false
  equippingOverhaulInstalled = false
  recommendedModCountExceeded = false
}
```

```

armors {
    armorRatingThresholds {
        heavy {
            body = 74
            feet = 27
            hands = 27
            head = 35
            shield = 54
        }
        light {
            body = 62
            feet = 18
            hands = 18
            head = 26
            shield = 44
        }
    }
}

```

FormIDs are specified in SkyProc's internal format that is independent of the actual load order. The FormID is specified by the 6-digit hexadecimal formID without load order index, followed by a space and the origin plugin name. As examples, 000007 Skyrim.esm references the player record and 2F389F Requiem.esp is Requiem's internal version stamp.

## Configuration Conflict Resolution

The Reqticator will load all configuration files that belong to Requiem.esp or any active mod which has Requiem.esp as a master. Conflict resolution happens on a key-by-key basis and the value from the last loaded mod wins, just like regular Skyrim conflict resolution.

✔ Your configuration file should only contain the settings you want to change.

All the other settings will then be taken from the base config provided by Requiem or the overwrites provided by other mods.

### Configurable Features

This section is a complete reference of all the available configuration options. The path-syntax used here is equivalent to nested object notation presented before.

## Handling the player record

Requiem applies a number of changes to the player record. For compatibility reasons these are not applied to the record directly but instead applied when the Requiem for the Indifferent.esp is created.

configuration key	data type	default value	description
playerRecord.healthOffset	integer	0	bonus health to add to player (can be negative)
playerRecord.magickaOffset	integer	0	bonus magicka to add to player (can be negative)
playerRecord.spellsToRemove	array of formIDs	[Flames, Healing]	spells to remove from the player character
playerRecord.staminaOffset	integer	0	bonus stamina to add to player (can be negative)

## Setup-related Warnings

Requiem makes a variety of sanity checks to minimize the probability of faulty installations. Some of these warnings are not always appropriate in the context of third-party extentions. Any non-critical warning the user can choose to ignore can also be suppressed by mods.

configuration key	data type	default value	description
ignoredWarnings.bashedPatchUsed	boolean	false	

			disable the warning shown when a Bashed Patch is found in the load order
<code>ignoredWarnings.bugFixesSksePluginMissing</code>	boolean	false	disable the warning shown when the Bug Fixes SKSE plugin is not found
<code>ignoredWarnings.cobbFixesSksePluginMissing</code>	boolean	false	disable the warning shown when the Cobb Bug Fixes SKSE plugin is not found
<code>ignoredWarnings.crashFixesSksePluginMissing</code>	boolean	false	disable the warning shown when the Crash Fixes SKSE plugin is not found
<code>ignoredWarnings.equippingOverhaulInstalled</code>	boolean	false	disable the warning shown when the Equipping Overhaul mod is found in the load order
<code>ignoredWarnings.recommendedModCountExceeded</code>	boolean	false	disable the warning shown when the load order contains more than 100 mods

## Thresholds for Armor-Rating Adjustments


Starting from version 4.0.0 Requiem supports final armor ratings in plugins. Every armor record that has an armor rating above a type-specific threshold will not have its armor rating adjusted by the Reqtificator. You can change the following settings to adjust the threshold for each item type individually.

configuration key	data type	default value	description
<code>armors.armorRatingThresholds.heavy.body</code>	int	74	armor rating threshold for heavy armors with the <code>ArmorCuirass</code> keyword
<code>armors.armorRatingThresholds.heavy.feet</code>	int	27	armor rating threshold for heavy armors with the <code>ArmorBoots</code> keyword
<code>armors.armorRatingThresholds.heavy.hands</code>	int	27	armor rating threshold for heavy armors with the <code>ArmorGauntlets</code> keyword
<code>armors.armorRatingThresholds.heavy.head</code>	int	35	armor rating threshold for heavy armors with the <code>ArmorHelmet</code> keyword
<code>armors.armorRatingThresholds.heavy.shield</code>	int	54	armor rating threshold for heavy armors with the <code>ArmorShield</code> keyword
<code>armors.armorRatingThresholds.light.body</code>	int	62	armor rating threshold for light armors with the <code>ArmorCuirass</code> keyword
<code>armors.armorRatingThresholds.light.feet</code>	int	18	armor rating threshold for light armors with the <code>ArmorBoots</code> keyword
<code>armors.armorRatingThresholds.light.hands</code>	int	18	armor rating threshold for light armors with the <code>ArmorGauntlets</code> keyword
<code>armors.armorRatingThresholds.light.head</code>	int	26	armor rating threshold for light armors with the <code>ArmorHelmet</code> keyword
<code>armors.armorRatingThresholds.light.shield</code>	int	44	armor rating threshold for light armors with the <code>ArmorShield</code> keyword

## Keyword Reference

### High-level description of the feature

Requiem adds a variety of keywords to the game. Some of them are used directly ingame for various Requiem features, while others are used to influence the behavior of the Reqtificator, our preprocessing tool and automated patch generator.

 This reference covers the keywords we consider the most relevant ones. If a keyword you're interested in is not listed here, please reach out to us to inquire about its usage.

### Detailed Documentation

- [Keywords used by ingame features](#)
- [Keywords used by the Reqtificator](#)
  - [Altering the Reqtificator's behavior on a per record level](#)
  - [Keywords used as identifiers for the automated rules](#)

## Keywords used by ingame features

These keywords are directly used by various ingame features of Requiem. Some of them are added via rules to the records when running the Reqtificator.


✔ “Distributed by Reqtificator” in the table below means that these keywords are usually added to the records by the Reqtificator. In cases where the rules defined by Requiem don't yield the desired outcome, you can always attach the desired keyword directly to a record to bypass a rule for this particular record.

keyword(s)	applied to...	description
REQ_AmmoWeight_...	ammunitions	defines the weight of the ammunition object, which is emulated via scripts
REQ_Armor_Resistance_... (distributed by the Reqtificator)	armors (cuirass)	defines the extra armor rating against specific damage types, most notably ranged attacks
REQ_ArmorPiercingArrow_Tier_...	arrows	defines the armor penetration power of the arrow, higher tiers are more powerful
REQ_ArmorPiercingBolt_Tier_..	bolts	defines the armor penetration power of the bolt, higher tiers are more powerful
REQ_DamageType_... (distributed by the Reqtificator)	weapons	defines the damage type of the weapon, used e.g. for creatures' natural resistances
REQ_Tempering_... (distributed by the Reqtificator)	weapons and armors	determines which smithing perk grants a tempering bonus for the item

## Keywords used by the Reqtificator

Altering the Reqtificator's behavior on a per record level

These keywords can be attached to individual records to change the Reqtificator's behavior for a single record.

keyword(s)	applied to...	description	
RFTI_Exclusions_NoArmorRatingRescale	armors	no changes to armor rating	
RFTI_Exclusions_NoBowSpeedRescale	ranged weapons	no changes to attack speed, no tagging as heavy bow/crossbow	
RFTI_Exclusions_NoDamageRescale	ammunition and weapons	no changes to damage	
RFTI_Exclusions_NoWeaponReachRescale	melee weapons	no changes to weapon reach	
<div> This deprecated keyword will be removed in a future release. Use the RFTI_Exclusions_... keywords instead.</div>	RFTI_DEPRECATED_AlreadyReqitized	armors	no changes to armor rating
		weapons	no changes to damage, melee weapon reach and ranged weapon attack, no tagging as heavy bow/crossbow for ranged weapons
		ammunition	no changes to damage

Keywords used as identifiers for the automated rules

The Reqtificator offers the possibility to define rules that automatically distribute additional keywords to weapons and armors based on the keywords they already have. A similar feature is available for actors to distribute perks and spells based on conditions.

The keywords listed below are used exclusively as input for these rules and have no direct ingame usage.

keyword(s)	applied to...	description
REQ_ArmorSet_..	armors	set identifiers used to distribute keywords automatically (e.g. for tempering), every armor is part of exactly one set
REQ_Cuirass_...	armors (cuirass)	identifier used to distinguish different items of the same armor set (e.g. vampire royal armor)



## Hands-On Tutorials


 We also have reference documentation available: [Reference Documentation](#)

- [Suppress the too many mods warning from the Reqtificator](#)

This hands-on tutorial shows how to configure the Reqtificator's features at the example of disabling a particular setup check during the installation process.

### Suppress the too many mods warning from the Reqtificator

This hands-on tutorial shows how to configure the Reqtificator's features at the example of disabling a particular setup check during the installation process.

 Check out these pages for the full reference documentation of the features presented in this tutorial:

- [Reqtificator Configuration](#)

### The problem we want to solve

Let's assume you work on a mod pack which results in having more than 100 active plugins in the default setup. The Reqtificator will then show a warning that it's not recommended to install Requiem on top of your existing load order and recommend to start your Requiem journey with a leaner load order. Especially users that are new to Requiem are often confused by this and you need to tell them that this warning is safe to ignore when they reach out for help. By adjusting the Reqtificator's configuration you can suppress this warning when your mod pack is installed and thereby streamline the installation process considerably.

### Step by step instructions

To make use of this feature, your mod needs to have a plugin, in this tutorial it will be called `Requiem Epic Improvements.esp`.

1. Add `Requiem.esp` as a master to your plugin if it doesn't already depend on Requiem.
2. Create the folder `SkyProc Patches\Requiem\Config\Requiem Epic Improvements` in your mod.
3. Use your favorite editor to create the new file `SkyProc Patches\Requiem\Config\Requiem Epic Improvements\Reqtificator.conf`
4. edit the content of the file you just created to contain the following content

```
ignoredWarnings {  
    recommendedModCountExceeded = true  
}
```

5. save the configuration file and test the installation of your mod pack on a fresh installation – the warning about the load order size will no longer appear

And that's already everything you need to do to suppress this Reqtificator warning. Check out the reference documentation if you want to know which other Reqtificator features you can adjust with this configuration file.

## Modding Resources

A collection of general modding resources that helpful when working with Requiem.

- [Artwork Resources for third-party authors](#) —  
This page contains a collection of Requiem-based artworks you can use to decorate your Requiem patches, Let's Plays and other Requiem-related content.
- [Delta Patches for Requiem releases](#) —  
Here you can find Delta patches for each Requiem release. These files contain all the records that have been added, removed or changed compared to the previous Requiem release.
- [EditorID conventions](#) —  
Many of the EditorIDs in Requiem follow a convention to improve the navigation through the plugin. The parts of the convention we consider mature and unlikely to change are documented here.























### Delta Patches for Requiem releases

Here you can find Delta patches for each Requiem release. These files contain all the records that have been added, removed or changed compared to the previous Requiem release.

Use these patches to quickly see what changed in a new Requiem release on a technical level and assess the changes that impact your own work.

We provide these delta patches for each new Requiem release compared to the previous release as a convenience for other mod authors. These files are created with the new delta patch function introduced in Tes5Edit version 4. If you need to “jump” over a release for some reason, you can thus easily create your own delta patch.

There are two versions of the delta patches: one uses the old version as the base plugin for the comparison (e.g. 2.0.2 vs 3.0.0) while the other uses the new version as the base plugin (e.g. 3.0.0 vs 2.0.2). The actual content is the same, but the representation differs a bit. Give them a try and find out which one you prefer to work with.

Old Version as Base Plugin		New Version as Base Plugin	
File	Modified	File	Modified
 Requiem 4.0.1 vs 4.0.2.esu	May 26, 2021 ogerboss	 Requiem 4.0.2 vs 4.0.1.esu	May 26, 2021 ogerboss
 Requiem 4.0.0 vs 4.0.1.esu	Jul 03, 2020 ogerboss	 Requiem 4.0.1 vs 4.0.0.esu	Jul 03, 2020 ogerboss
 Requiem 3.4.0 vs 4.0.0.esu	Jul 01, 2020 ogerboss	 Requiem 4.0.0 vs 3.4.0.esu	Jul 01, 2020 ogerboss
 Requiem 3.3.0 vs 3.4.0.esu	Dec 29, 2019 ogerboss	 Requiem 3.4.0 vs 3.3.0.esu	Dec 29, 2019 ogerboss
 Requiem 3.2.0 vs 3.3.0.esu	Oct 05, 2019 ogerboss	 Requiem 3.3.0 vs 3.2.0.esu	Oct 05, 2019 ogerboss
 Requiem 3.1.1 vs 3.2.0.esu	Aug 18, 2019 ogerboss	 Requiem 3.2.0 vs 3.1.1.esu	Aug 18, 2019 ogerboss
 Requiem 3.1.0 vs 3.1.1.esu	Jul 13, 2019 ogerboss	 Requiem 3.1.1 vs 3.1.0.esu	Jul 13, 2019 ogerboss
 Requiem 3.0.2 vs 3.1.0.esu	Jul 04, 2019 ogerboss	 Requiem 3.1.0 vs 3.0.2.esu	Jul 04, 2019 ogerboss
 Requiem 3.0.1 vs 3.0.2.esu	Apr 12, 2019 ogerboss	 Requiem 3.0.2 vs 3.0.1.esu	Apr 12, 2019 ogerboss
 Requiem 3.0.0 vs 3.0.1.esu	Mar 16, 2019 ogerboss	 Requiem 3.0.1 vs 3.0.0.esu	Mar 16, 2019 ogerboss
 Requiem 2.0.2 vs 3.0.0.esu	Mar 04, 2019 ogerboss	 Requiem 3.0.0 vs 2.0.2.esu	Mar 04, 2019 ogerboss


 [Download All](#)

 [Download All](#)

### Artwork Resources for third-party authors

This page contains a collection of Requiem-based artworks you can use to decorate your Requiem patches, Let's Plays and other Requiem-related content.

The assets on this page have been created by Vallen128 with contributions from Lazuri and Metaseverity

 You may use all artwork on this page in your Requiem-derived works without asking for explicit permission from us. If you wish to use any Requiem artwork not found on this page, please contact us via private message at NexusMods.

### Compatibility Patches





## Let's Plays

### EditorID conventions

Many of the EditorIDs in Requiem follow a convention to improve the navigation through the plugin. The parts of the convention we consider mature and unlikely to change are documented here.

Each and every record added by Requiem should have the prefix `REQ_`. Records that are used or distributed by the Reqtificator have the prefix `RFTI_` instead.

- [Special Prefixes \(all records\)](#)



- [Container \(CONT records\)](#)
- [Actors \(NPC\\_ records\)](#)
- [Perks \(PERK records\)](#)
- [Weapon \(WEAP records\)](#)

### Special Prefixes (all records)

- [Ingestible \(ALCH records\)](#)
- [Armor, Clothing and Jewelry \(ARMO records\)](#)





Record description	EditorID	Examples
records used solely for maintaining backwards compatibility, will be deleted in next breaking release	REQ_DEPRECATED	RFTI_DEPRECATED_AlreadyReqtified
records used solely for maintaining backwards compatibility, still in use and may not be deleted	REQ_LEGACY	REQ_LEGACY_MageArmor50
vanilla records that are completely unused in requiem	REQ_NULL	REQ_NULL_AgileDefender40

Ingestible (ALCH records)

Record description	EditorID	Examples
drink	REQ_Drink_<name>	REQ_Drink_Ale

food	REQ_Food_<name>	REQ_Food_CookedMeat
poison	REQ_Poison_<effect>[<tier>]	REQ_Poison_DamageHealth01
potion	REQ_Potion_<effect>[<tier>]	REQ_Potion_RestoreHealth01

#### Armor, Clothing and Jewelry (ARMO records)

Record description	EditorID	Examples
standard armor or clothing	REQ_<type>_<set>_<slot>	REQ_Cloth_Jester_Head REQ_Heavy_Ebony_Body REQ_Heavy_Dragonplate_Shield REQ_Light_Forsworn_Feet REQ_Light_Hide_Hands
standard jewelry	REQ_<slot>_<set>	REQ_Circlet_SilverMoonstone REQ_Amulet_AncientNord REQ_Ring_Gold
(generic) enchanted version of a standard armor or clothing	REQ_Ench_<type>_<set>_<slot>_<enchantment>[<tier>]	REQ_Ench_Heavy_Ebony_Body_Alteration3
(generic) enchanted version of standard jewelry	REQ_Ench_<slot>_<set>_<enchantment>[<tier>]	REQ_Ench_Ring_Silver_Illumination
unique variant of a standard armor or clothing	REQ_Var_<type>_<set>_<slot>_<name>	REQ_Var_Cloth_Jester_Head_Cicero REQ_Var_Heavy_Ebony_Body_Irileth REQ_Var_Light_Forsworn_Feet_OldGods
unique variant of standard jewelry	REQ_Var_<slot>_<set>_<name>	REQ_Var_Circlet_SilverMoonstone_MG04Reward REQ_Var_Amulet_AncientNord_Saarthal REQ_Var_Ring_Gold_Asgeir
non-playable version of a standard armor or clothing	REQ_NP_<type>_<set>_<slot>	REQ_NP_Heavy_Dragonplate_Shield REQ_NP_Light_Hide_Hands
non-playable version of standard jewelry	REQ_NP_<slot>_<set>_<name>	REQ_NP_Amulet_AmuletOfArticulation
artifact	REQ_Artifact_<name>	REQ_Artifact_AetherialCrown
creature armor or skin	REQ_Creature_<creature>_<name>	REQ_Creature_VampireLord_Ornament
saddle	REQ_Saddle_<set>	REQ_Saddle_Daedric

#### Container (CONT records)

Record description	EditorID	Examples
merchant chests, in case of name clashes shop name/owner should be used instead of location	REQ_VendorChest_<shop type>_<location>	REQ_VendorChest_Apothecary_Markarth  REQ_VendorChest_General_Riften  REQ_VendorChest_General_BrandShei



## Actors (NPC\_records)

Record description	EditorID	Example
template for other bandit records, not functional on its own	REQ_Bandit_Template_<class>	REQ_Bandit_Template_AxeShield_01
generic bandit, functional on its own	REQ_Bandit_Generic_<class>	REQ_Bandit_Generic_1H
bandit with custom AI data to only attack when enemies come too close	REQ_Bandit_Aggro<Warn/Attack Distance>_<class>	REQ_Bandit_Aggro1024_Melee
bandit with custom AI data, AI packages and scripts to ambush the player	REQ_Bandit_Ambush_<class>	REQ_Bandit_Ambush_Missile
bandit with custom AI packages to guard something	REQ_Bandit_Guard_<class>	REQ_Bandit_Guard_2H
bandit with a custom combat override package list	REQ_Bandit_HoldPosition_<class>_<package>	REQ_Bandit_HoldPosition_Missile_CurrentLoc256
bandit with custom properties for a specific location	REQ_Bandit_Loc_<location>_<use case>	REQ_Bandit_Loc_DaintySload_1H REQ_Bandit_Loc_ValtheimTowers_Elsi REQ_Bandit_Loc_WhiteRiverWatch_Alarm1
look template for ActorVariants	REQ_LookTemplate_<vanilla EditorID>	REQ_LookTemplate_EncBandit01MagicArgonianM

## Perks (PERK records)

Record patterns with a light blue background should not be assigned directly to NPCs in plugins.

Record description	EditorID	Examples
standard perks that can be picked by leveling up	REQ_<skill>_<perk>	REQ_Destruction_FrostMastery
perks that mimic inherent (racial) abilities which cannot be modelled as constant spells	REQ_Trait_<creature>	REQ_Trait_Ghost
perks that provide functionality for abilities that cannot be modelled as magic effects	REQ_Ability_<ability>_<effect>	REQ_Ability_BendTheLawOfFirsts_AllowTwoEnchantments
perks that provide functionality for potions that cannot be modelled as magic effects	REQ_Alch_<potion>_<effect>	
perks that provide functionality for enchantments that cannot be modelled as magic effects	REQ_Ench_<ench>_<effect>	REQ_Ench_Soulreaping_ReducedHealing
perks that provide functionality for food that cannot be modelled as magic effects	REQ_Food_<food>_<effect>	
perks that provide functionality for powers that cannot be modelled as magic effects	REQ_Power_<power>_<effect>	REQ_Power_OrcBerserk_DamageBonuses
perks that provide functionality for scrolls that cannot be modelled as magic effects	REQ_Scroll_<scroll>_<effect>	REQ_Scroll_SoulGemEvocation_NoXPGain
perks that provide functionality for spells that cannot be modelled as magic effects	REQ_Spell_<spell>_<effect>	REQ_Spell_FeatherFalling_NoFallingDamage
perks that are attached to the player record	REQ_Player_<feature>	REQ_Player_AllowShouting
perks that are distributed by the Reqtificator to all NPCs and the player	RFTI_All_<feature>	RFTI_All_ActorValueModifier
perks that are distributed by the Reqtificator to all NPCs but not the player	RFTI_NPC_<feature>	
perks that are distributed by the Reqtificator to the player	RFTI_Player_<feature>	RFTI_Player_ExperienceAdjustment

## Weapon (WEAP records)

Record description	EditorID	Examples
standard weapon	REQ_<set>_<type>	REQ_Iron_Waraxe
standard magic staff	REQ_Staff_<set>_<enchantment>	REQ_Staff_Alteration_Magelight REQ_Staff_Falmer_LightningBolt
(generic) enchanted version of a standard weapon	REQ_Ench_<set>_<type>_<enchantment>	REQ_Ench_Iron_Waraxe_DrainHealth
unique variant of a standard weapon	REQ_Var_<set>_<type>_<name>	REQ_Var_Glass_Greatsword_Grimsever
non-playable version of a standard weapon	REQ_NP_<set>_<type>	REQ_NP_Daedric_Battleaxe
artifact	REQ_Artifact_<type>_<name>	REQ_Artifact_Staff_Aetherial REQ_Artifact_Sword_Dawnbreaker
non-playable weapon used by a creature	REQ_Creature_<creature>_<type>	REQ_Creature_Giant_Club
weapon that is none of the above	REQ_Special_<type>	REQ_Special_Pickaxe REQ_Special_Staff_SpiderControlRod