Manual for Malevolent Mobs (version 0.4.3)

Contents

[Installation and first run 3](#_Toc443606223)

[General configuration 4](#_Toc443606224)

[General 4](#_Toc443606225)

[Spawns 6](#_Toc443606226)

[Malevolent Mobs 8](#_Toc443606227)

[Core premise of each mob 8](#_Toc443606228)

[Standards for the mobs 8](#_Toc443606229)

[General behavior of the mobs 9](#_Toc443606230)

[Rewards given upon death 9](#_Toc443606231)

[Experience 9](#_Toc443606232)

[Items 10](#_Toc443606233)

[Damage model 11](#_Toc443606234)

[Armor 11](#_Toc443606235)

[Order of abilities 11](#_Toc443606236)

[The role of item effects 12](#_Toc443606237)

[Explosions and lightning strikes 12](#_Toc443606238)

[Types of damage 13](#_Toc443606239)

[Abilities 14](#_Toc443606240)

[Passive abilities 14](#_Toc443606241)

[Active abilitives 18](#_Toc443606242)

[Roles 19](#_Toc443606243)

[Item effects 22](#_Toc443606244)

[Loot and rewards 31](#_Toc443606245)

[Permissions 32](#_Toc443606246)

[Commands 33](#_Toc443606247)

[Debugging 34](#_Toc443606248)

# Installation and first run

The plugin is to be added to the “plugin” folder as usual. When the plugin is first run, it will automatically generate a new main configuration file, in addition to many other files. This main configuration file will never be modified on plugin startup again; to regenerate this file, the file has to be deleted or renamed. It’s possible that the configuration files are changed during normal use, as there is some data used in this plugin that must remain persistent. Expect the plugin to write the configuration files to disk every time the server is stopped, or whenever a user saves the configuration files using the plugin commands to do so.

# General configuration

There is a huge amount of settings to tweak, modify or otherwise work with in this plugin. In this section, only the most basic and essential settings will be discussed; the rest will be covered in other sections of this manual.

The configuration file is split up into multiple sections. The most essential one is the section named *General*, and this is where the settings that directly affect the behavior and performance of the plugin can be found. This section is also split up into multiple sub-sections.

## General

*debugMode*: If this setting is set to true, the plugin will output many different messages to the console, and alter some mob spawning behavior; it is not recommended to have this value set to *true* unless debugging is needed. Default value: *false*

#### Bossbar

*mobHealthbarDistance*: The distance a player has to be from a Malevolent Mob in order to see their health bar. If the distance to the mob is greater than this number, the health bar is removed. The distance is given in meters (number of blocks). Default value: *80*

*mobHealthbarUpdateTime*: The time between each time the health bar is updated for all players. The higher this number is, the less the drag on the server is, but the slower the health bar will reach when the Malevolent Mob is attacked or healed. Default value: *10*

#### Mobs

*maxLevel*: The highest level a Malevolent Mob can have by normal means of spawning. A higher level mob is more difficult to kill than a lower level mob. The difficulty is not necessarily increasing linearly with level. Default value: *50*  
Note that the plugin is by default balanced with a maximum level of *50*. A maximum level much higher than this is not recommended, if using otherwise default settings.

*levelSearchDistance*: The Malevolent Mobs scale with the player levels (if SkillAPI is installed), where the player level is defined as the highest level of all the classes the player is professed as. This setting tells the mobs how close a player has to be in order to be considered when calculating the mob level. Default value: *150*

*minSpawnDistance*: The shortest distance a Malevolent Mob can have to a player when spawned. If the distance to a mob that would spawn to the closest player is shorter than this, the spawn is cancelled. This does not apply to mobs that are spawned with commands or the dungeon system, or if the plugin is running in debug mode. Default value: *20*

*healthScale*: The health of a Malevolent Mob is multiplied with this number. Default value: *4.0*  
Note that the plugin has been balanced with this number being *4*. Using high numbers will make the fights last longer, but also potentially more dragged out than desired. Using too small numbers will make the fights highly in favor of the players, as well as making the fights potentially much shorter than desired.

*damageScale*: The damage that is dealt by the mob from abilities will be multiplied by this factor. This applies to both active and passive abilities, including auras that deal damage over time. This will not change the damage dealt by effects such as wither, fire, poison or similar effects. Default value: *1*Note that the plugin has been balanced with this number being *1*. Using too high numbers is likely going to lead to players being one-shotted by the Malevolent Mobs in one round of abilities, while using too low numbers if likely to lead to insignificant damage output from the mobs. The damage due to effects such as wither, poison and fire will remain unchanged.

*armorScale*: An Malevolent Mob has a some armor values; the amount of armor the mobs have will be scaled by this factor. Default value: *1*  
Note that the plugin has been balanced with this number being *1*. Using high numbers is likely to yield a large quantity of armor to mobs, leading to players dealing less physical, magical and projectile damage to Malevolent Mobs, but the damage due to effects such as wither, poison and fire will remain unchanged.

*targetAquireDistance*: The distance a Malevolent Mob must have to a player or a monster (that hasn’t been targeted yet) to be considered a valid target. The current target of the mob will always be prioritized first. Default value: *25*

*targetLostDistance*: The distance a Malevolent Mob must have to a player or monster (that has already been targeted) to be considered an invalid target. The current target of the mob will never be lost, unless that mob lost agro due to normal Minecraft behavior. Default value: *75*

*base/scaleVanillaExperience*: The amount of regular experience the Malevolent Mob provides on death. On the most basic level, the experience amount is calculated as base + level \* scale. If multiple people are participating in the combat, the experience is shared amongst every participant based on how much damage they dealt to the mob. The higher fraction of the total damage a player dealt, the more experience that player is rewarded. Default values: *base 100, scale 8*

*base/scaleSkillAPIExperience:* The amount of SkillAPI class experience that the Malevolent Mob provides on death. The amount is calculated the same way as normal Minecraft experience is calculated. Default values: *base 75, scale 5*

#### Attacks

*treatAttackAsRangedDistance*: The distance a Malevolent Mob must have to an attacking player for the player’s attack to count as ranged. Attacks that are treated as ranged will be further reduced in strength by the mob’s defenses. Default value: *15*

*treatAttackAsMagicDistance*: The distance a Malevolent Mob must have to an attacking player for the player’s attack to count as magic. If *treatSkillsAsMagic* is *false*, this setting has no effect. Default value: *5*

*treatSkillsAsMagic*: If true, player’s SkillAPI skills will be treated as magic for all intents and purposes against Malevolent Mobs, provided the players are more than *treatAttackAsMagicDistance* meters away from the target they are attacking. The players’ skills will also be treated as magic damage if used against players, if this setting is set to *true*. Default value: *true*

*attackCooldown*: The default base attack cooldown for all Malevolent Mobs, in seconds. Most mobs override this number, but if not otherwise specified, the mobs will default to this many seconds of delay between the active abilities they use. Default value: *3*

#### Defense

*armorEfficency*: Whenever a player is attacked by a Malevolent Mob using a certain ability (for example a fire-based attack), all enchantment levels that work as defense against that attack are added up. The total armor value of the enchantments is given as the sum of the levels multiplied by the *armorEfficency* setting. Default value: *10*

*firePotionEnchantLevelEquivalent*: If the player has a the fire resistance status effect, this setting tells how many levels the effect would be if it had been an enchantment. Default value: *7*

*resistancePotionEnchantLevelEquivalent*: If the player has the damage reduction status effect, this setting tells how many levels the effect would be if it had been an enchantment, per strength level of the effect. Default value: *4*

## Spawns

In order to see Malevolent Mobs in the world, they have to be spawned in one way or another. While there are several ways to spawn them in, the easiest is to let the plugin itself spawn them in. This section has all the information required to deal with the natural spawning of mobs.

*allow*: This flag holds the information if mobs are allowed to be Malevolent on spawn or not. If this setting is *false*, no mob will ever spawn naturally – it is still possible to spawn mobs using the dungeon system and spawn commands, though. Default value: *true*

*spawnChance*: The chance of a normal mob to become Malevolent when it is spawned in the world, regardless of how it was spawned (exception: no normal mob spawned by the Malevolent Mobs’ abilities or by the dungeon system will ever be Malevolent). Default value: *0.005 (0.5%)*

*blockedWorlds*: This setting holds a list of all the worlds Malevolent Mobs are not allowed to spawn in naturally. Any Malevolent Mob that is spawned via a command, ability, or via the dungeon system will spawn as normal, even in worlds that are included in this list. The world names are case-sensitive.

#### Types of Malevolent Mobs

In addition to these general parameters, it’s possible to tweak the spawns somewhat on a per-mob basis. Any mob that is not listed in this section will never spawn naturally, no exceptions. For those that are listed, however, they can be enabled or disabled from spawning naturally. They can also have a different chance of spawning, which can be useful for making mobs of one type more rare or common than another.

In theory, any entity can be Malevolent. However, it is not recommended to attempt to make anything but living entities (any animal or monster) Malevolent. The behavior of Malevolent entities is undefined if the entity is not an animal or monster. All enum values under

https://github.com/Bukkit/Bukkit/blob/master/src/main/java/org/bukkit/entity/EntityType.java

are valid entity names, in addition to a few others. If an entity is not specified there, it is *not* valid. The capitalization of the name matters; by convention, the first letter in every word has to be upper-case. For example, *Zombie*, *Skeleton* and *Cave Spider* are all valid, but *enderman*, *SKELETON* and *Zombie pigman* are all invalid. The system doesn’t differentiate between underscores or spaces, so either works.

Spawns:

Enable: true

spawnChance: 0.015

Zombie:

allow: true

chanceMultiplier: 1.0

Spider:

allow: true

chanceMultiplier: 0.8

To enable a mob, the key *enable* has to be set to *true*, and the *chanceMultiplier* key has to be something larger than *0*. The key *chanceMultiplier* can be larger than *1*.

Example 1: How to enable natural spawns for zombies and spiders. In this example, the spiders have a lower chance (20% lower) of spawning than the zombies do. Zombies have 1.5% chance of spawning in this case.

#### Settings the roles

Just enabling the spawn for mobs isn’t going to grant them any behavior. It’s very important to tell the plugin what *role* a Malevolent Mob can have when it first spawns.

*roles*: A list of all the roles that the mob of the specified type can have. All entries in this list should follow standard name conventions, where the first letter of each word is capitalized.

Spawns:

Skeleton:

roles:

- Archer

- Ancient Warrior

- Necromancer

If no role is specified, the behavior of the Malevolent Mob is undefined and may cause issues. The plugin will attempt to resolve this issue by removing any Malevolent Mob that doesn’t have any role defined. If a non-existing or invalid role is selected and the mob attempts to load up that role, the plugin will detect this and select another role from the list for as long as that is possible.

Example 2: Assign some roles to skeletons.

# Malevolent Mobs

The entire plugin is revolved around one single aspect, and that is providing some mobs that are tough, hard to kill and are a challenge for players to deal with. The mobs are granted special abilities and defenses in order to make this possible, with abilities ranging from launching a fireball to striking down the foes with bolts of lightning from the skies.

## Core premise of each mob

Each and every one of the default mobs are designed to fit within some role of what their natural abilities are. Zombies are seen as the main tough, stupid brute that uses raw force and durability to bring their foes down. Skeletons are the weaker, but faster and more agile buggers that fights either from a distance with a bow or up close and personal with a sword. Other mobs have their own special abilities and attacks which the default Malevolent Mobs will attempt to strengthen and expand upon.

Of course, while each mob is designed to kill their targets, each and every one of them has been granted at least one form of weakness, as well as making sure that all their abilities and attacks can be countered, dodged or otherwise avoided. This is one of the most fundamental design choices that has gone into the making of this plugin, and it can be seen in each and every aspect – when the player dies, it should feel like it’s the player’s fault (*I should have dodged that fireball*, *perhaps it would have been better to walk behind that wall instead of trying to drink that potion of health*).

The mobs themselves are designed to deal minimum damage in one combo, but to maintain a high damage over time. This is a critical design choice which allows players a chance to retaliate or evade the encounter should they find themselves unable to take the fight there and then.

## Standards for the mobs

When it comes to what the mobs can do and what they can’t, it all boils down to their abilities and stats. In general, the mobs are set to have relatively weak stats at low levels – often stats such that a single low-level player (regardless of class) can kill the mob alone. As the mob grows in levels, it will become tougher, eventually reaching the point where a single player can’t easily kill it alone anymore, regardless of equipment and single-use items. This should be when the mob’s level is around half the maximum level. When the mob is close to the maximum level, it should be a challenge for even a team of several organized people to take it out.

The ability distribution for the mobs as function of levels is highly dependent upon the mob itself and what kind of abilities they have. For high-damage-dealing mobs, they should have few abilities in the beginning, with one introduced every four-five levels or so. For those that have several abilities that have a chance of appearing, the abilities with the worst effects should only appear once the mobs reach a certain level (as in, having zero chance of appearing before the nth level).

For the default mobs, the standard ability distribution is ten abilities per role, five of them passive and five of the active abilities. The distribution over levels varies, but in general all roles have their full skillset at around level *30*, and around half of their abilities at levels lower than *8*. The strength, range and duration of the abilities will increase with a higher level, but rarely if ever by more than double that of the values at level one (even less for the damage parameters). This is yet another important aspect to the core of the plugin – the mobs shall not be able to one-shot players, or kill them in one rotation of abilities. Instead, to make a mob more challenging, new abilities are introduced with higher levels, and the cooldown of the abilities can be reduced with increasing levels.

In terms of health and other stats, the Malevolent Mobs’ base for health is some percent greater than what the normal health of the corresponding mob – in some cases, it might even be lower (note that this is before factoring in the health multiplier defined in the *General* section of the configuration file). The health of the mobs will increase with level, in order to keep up with the damage output from players of higher levels. Generally, the health at maximum level is usually between two-three times the health at level one.

When it comes to the stats such as armor, this is highly based on both the role of the mob, the base type of the mob as well as the desired weaknesses and strengths of the mob. In general, the base values are relatively low, but they increase very quickly with levels, often having armor levels equal to double the base values at level eight or less and several hundred at maximum level.

## General behavior of the mobs

The Malevolent Mobs will be based highly on normal Minecraft mob logic; this is completely unchanged in this plugin. However, there are some additional behaviors that worth noting. The mobs will not seek out only one target, if there are multiple people nearby.

The target the mob is trying to actively attack (the main target) will be subjected to the main bulk of the active abilities the mob has. If an active ability couldn’t be used on the main target, the ability will be attempted to be used on the secondary targets instead.

As the health of the mob is decreasing, the mob may introduce other abilities (depends on the skillset of the mob). For example, certain abilities trigger only when the health of the mob is below half of the maximum health.

// TODO: Add what the mob does in the case of drowning, burning, suffocation etc when it has been coded into the plugin

## Rewards given upon death

Whenever a player is capable of slaying a Malevolent Mob, they are granted a few types of rewards. The rewards come in two categories; experience and items. The rewards are based only on the level of the mob, where a higher level means a higher bonus.

// TODO: Write about the role and custom drops when that has been coded in

### Experience

A mob that is attacked will keep track of the damage dealt to it, and the players whom dealt the damage. Natural damage, such as fire damage, fall damage, suffocation and such are also tracked. Over time, some of the damage tracked is removed from each source, meaning that if a player did only a tiny bit of damage in the beginning, that counts for nothing at all in the end of the fight, if the player didn’t actively fight the mob after that initial attack.

When the mob dies, the players who participated in the combat is granted an amount of experience equal to the total value of experience multiplied by their participation factor, which is defined as the damage they dealt divided by the total damage received by the mob. A consequence of this is that players who kill the mob by means of suffocation, fall damage or other natural sources will receive little to none experience, while those that kill it with swords, bows, abilities or other similar means will receive close to the full reward.

### Items

Regardless of who and what dealt damage to the Malevolent Mob, it will (usually) drop an item upon death. The item dropped can be anything from a golden apple to a diamond axe; the item will also have a custom name and potentially one or more custom item effects that can be a wide number of things. More will be written about these effects in a different section of this manual.

# Damage model

In order to provide a good Malevolent Mob for the players to fight, it is crucial to understand the exact mechanisms behind the damage that is dealt, both to the mob as well as the player. The order of abilities is also something that should be considered, as certain abilities may affect the outcome of another if they are chained in a certain way.

It’s also important to consider what a player may use against the mob, and what the player may have at the time of fighting the mob, when trying to model how much damage the mob is capable of doing, and how much damage the player can do to the mob.

## Armor

The most basic but also most valuable defense of all the mobs is their armor value. A higher armor means that a higher proportion of damage is blocked, but all the damage will never be blocked. The possible theoretical amount of total armor for a mob is given in the range from -50 to *+*, with the damage percentage that is blocked by the armor given as

The damage that gets through the armor is given as

For an armor value of *-50* this yields an armor rating of *-1*, which means that the damage taken is doubled. If the armor value is *100*, the armor rating is *0.5* which means that only half of the damage gets through. A typical armor value for a level *50* Malevolent Mob is around *600*, which results in a damage reduction of about *85%*.

If players use melee attacks against the Malevolent Mobs, the mobs’ melee armor is used to compute the armor rating. If the players use a bow or magic, the armor rating is computed from the arrow or magic armor values, respectively. If the player is also a certain distance away from the mob (as specified by the *treatAttackAsRangedDistance* key under the *General* section in the configuration file), however, the mobs’ ranged armor value is added. This means a mob can get more than *1000* as armor value against the players’ attacks, resulting in an armor rating of more than *0.9*, meaning more than *90%* of the incoming damage is blocked.

The armor concept applies not only to the Malevolent Mobs, but also to players. Whenever a player is attacked by the Malevolent Mobs from certain sources (mostly fire and explosions), the player’s armor level is defined as the sum of the levels of the relevant enchantments and the effects of any status effects, multiplied by a constant factor (*armorEfficiency* which is defined in the *General* section). The armor rating of the players’ armor items does not count in the calculations against physical damages, as they are applied separately by Minecraft at a different point in time.

## Order of abilities

Whenever a Malevolent Mob use some sort of ability, be it passive or active, they are applied in a specific order. While this is more relevant for an attacking player, it is important to know what happens from the mobs’ perspective as well.

All abilities have a field which is their priority. Abilities with a low priority is applied before armor reduced the damage dealt (applies only to Malevolent Mobs, not players), while one with a normal or high priority deals the damage after this reduction, with the high priority applied last. This may seem counter-intuitive, but remember that the final ability may cancel the effects from another ability which has a lower priority. For example, the passive abilities *Exoskeleton* and *Guardian Angel* have two very different mechanisms of keeping the mobs alive, where one simply block damage and the other revive the mob if it dies. Here, the latter has a high priority and the former a normal priority, which means that the exoskeleton is guaranteed to be shattered before the guardian angel ability triggers.

When the mobs are attacking, the situation is somewhat changed. Mobs are not allowed to attack more than once with an active ability every so often, which means that the higher-priority abilities are performed first, then the lower-priority ones. The player damage reduction take place on the damage received and is irrelevant of the order of the abilities performed by the mobs. Similarly, when a mob is healed, abilities with a high priority take precedence.

## The role of item effects

There are certain item enhancers packed with the plugin, which allows some special effects and interactions between the Malevolent Mobs and players.

Some items may increase the potency of the weapon the player is using, reduce the armor of the Malevolent Mob or apply some form of status effect. All these item effects will apply before the damage to the mob is applied, and all damage modifiers on the item effects will apply before any of the abilities or stats of the mob can reduce the damage.

As for the effects of the items, the order they apply their effects in are specifically stated on the item they are attached to. For example, if the first effect is increasing the damage by three percent of the target’s current health, the second adds a flat one heart more of damage and the third increase the damage by ten percent, the damage *from the two first effects are also increased by ten percent*. Had the ten percent increase been the first effect, then the second two effects *would not have been increased* by the percentage.

## Explosions and lightning strikes

There are many abilities that rely on explosions or lightning strikes. These will not behave in the same way as Minecraft explosions and lightning strikes. Every explosion and lightning strike that is caused by this plugin will run custom logic, with the sole purpose of reducing the chance of accidental griefing to as close to zero as possible.

As such, explosions and lightning strikes will never deal damage to anything other than players, unless otherwise specified by the abilities themselves (which can be configured). Explosions will never destroy blocks and lightning strikes will never set blocks on fire.

Explosions will behave differently from Minecraft explosions; while the damage drops off as a function of distance to the center of the explosion, it drops off as a cosine function. This means that the damage is mainly concentrated in a somewhat wide circle around the center, and then quickly drops to zero at the edges. As for lightning strikes, they will always strike the topmost entity, or the topmost block, depending on which is highest up. The lightning bolts will never strike the inside of a cave, or a player that is safe under a roof (unless the roof is very thin). The damage dealt by a lightning strike is constant in the area of effect, and the targets will also be lit on fire for a short duration.

## Types of damage

// TODO: Write about physical, magical, natural (suffocation, fall, etc) and effect-based (wither, poison, etc) damage  
// TODO: Write about the difference between melee, ranged, magic and arrow (projectile) armor

# Abilities

The Malevolent Mobs have, as previously mentioned, several abilities they can use to kill their foes. These abilities can be sorted into two main categories, passive and active abilities. The passive abilities usually take effect every tick, when receiving or dealing damage, and/or when spawning/dying. The active abilities have a cooldown and the mobs are not allowed to use more than one active every few seconds.

All abilities will have some parameters in common, and there are some rules that govern how these parameters are chosen if they are not specified in a role. The parameters that are the same for all abilities are the following:

*enable*: A flag that tells if the ability is enabled or not; if disabled, no mob will ever be able to gain this ability, no exceptions  
*base/scaleChance*: The chance of a mob receiving this ability, where any value less than *0* means no chance of getting the ability and all values above *1* is guaranteed to give the mob the ability. Any values in-between are intermediate chances, where *0.7* is 70% chance of receiving the ability  
*base/scaleMaxChance*: If defined, this will mark the upper limit of the chance for a mob to have the given ability. This can be used to grant a Malevolent Mob a 75% chance of getting some ability, while also not granting the ability before a certain level (Example: base=-12, scale=1, baseMax=0.8 will yield a chance of 0.8 of getting the ability at level 13 and above, 0 at all other levels)

When adding abilities to a mob, note that any parameter that is not specified will pick its value from the defaults that are specified under the *Abilities* folder. In addition to this, any parameter that starts with *base* or *scale* will have a default value of *0*, with the exception of *baseChance* and *baseMaxChance* parameters, both which will be defaulted to *1*, if they aren’t specified.

## Passive abilities

Passive abilities can be split into a few categories themselves. The main groups are the auras or area of effect abilities, the on-damage abilities and the on-tick ones. Auras or AoE abilities will affect a certain area around the mob or a certain area, either by dealing damage to the targets in the vicinity or applying various effects. On-damage abilities apply some form of effect when the mob is attacked or is attacking, which can either reduce the damage the mob takes, enhance the damage dealt to the targets or anything else; shields usually fall under this category. On-tick passives usually apply some form of effect every nth tick or so, with effects ranging from regenerating some health to spawning some minions.

#### Deteriorating Aura

The deteriorating aura prevents any nearby players from being able to gain health from any regeneration effect or from natural healing. In effect, this means that the only ways for a player to gain more health is by using a potion of healing, move away from the mob, or use some ability that returns health. This aura is visualized as a cloud of red particles around the mob.

It is recommended that this ability is never used together with the *vampirism* ability, as that would make it impossible for the player to regain health while in combat with the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the aura, how far it can reach from the center of the mob

#### Exoskeleton

A mob with the exoskeleton ability will have an additional layer of defense which must be breached in order to damage the mob. When the exoskeleton has some charge left, all the damage dealt to the mob will be mitigated to the shield until it shatters. If the mob isn’t attacked for some time, the exoskeleton grows back. Attacking the mob will disable any current growth. When a mob with this ability is attacked, a sound (anvil hitting the ground) will play every time this ability takes effect.

**Priority**: Normal **Parameters**:  
*base/scaleRegeneration*: How many points the exoskeleton restores each time it recharges  
*base/scaleStrength*: The minimum amount of damage that needs to be dealt to the mob to start damage it (Strength of the exoskeleton)  
base/scaleStartupDelay: The number of ticks for which the mob must not be attacked in order to start recharging the exoskeleton  
*base/scaleRepeatDelay*: The number of ticks between each successive recharge

#### Explosion

The explosion ability will cause the mob to explode when it dies, severely damaging any target in the vicinity. When a mob has this ability and is about to die, a creeper hissing sound will play.

**Priority**: Normal **Parameters**:  
*base/scaleStrength*: The amount of damage that the explosion will deal in the center of the blast  
*base/scaleRadius*: The size of the explosion, how far it can reach from the center  
*affectPlayersOnly*: Indicates if only players can be damaged by the explosion, or if all entities can be damaged by it

#### Fire Aura

The fire aura is an aura that will set any entity within it on fire for as long as the entity remains inside the aura. As soon as they move out, the fire will extinguish unless they have another active fire effect that lasts for a longer amount of time. This aura is visualized as a cloud of fire particles around the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the aura, how far it can reach from the center of the mob  
*affectPlayersOnly*: Indicates if only players can be damaged by the aura, or if all entities can be damaged by it

#### Freezing Aura

The freezing aura is an aura that will slow any entity within range, and apply damage at a slower rate than other common auras. As soon as the entity steps out of reach of the aura, the slow effect will be removed unless the entity have another active slow effect that lasts for a longer amount of time. This aura is visualized as a cloud of snowball particles around the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the aura, how far it can reach from the center of the mob  
*base/scaleStrength*: The strength of the slow effect entities inside the reach of the aura receives  
*base/scaleDamage*: The amount of damage dealt to all entities within the reach of the aura  
base/scaleDelay: The number of seconds before the frost damage is applied to targets within the aura; this damage will be reapplied after this many seconds for as long as the effect is on a mob  
*affectPlayersOnly*: Indicates if only players can be damaged by the aura, or if all entities can be damaged by it

#### Guardian Angel

Whenever a Malevolent Mob receives a damage that is high enough to kill them, they will have their health reset back to the maximum health. Whenever this happens, one charge of the guardian angel ability is lost; once there are no more charges left, the mob will die for good when fatal damage is received. This aura is visualized as a cloud of white particles around the mob.

Care should be taken to avoid giving too many extra lives to a Malevolent Mob, as fights that lasts for a very long time could quickly get boring and repetitive.

**Priority**: High **Parameters**:  
*base/scaleLives*: The amount of extra lives for the mob

#### Mother

A mob that has the mother ability will spawn a given number of mobs of the same type at the mother’s location. The mother will never produce Malevolent Mobs, and it will not spawn more if the number of mobs in the nearby region is too large.

**Priority**: Normal **Parameters**:  
*minGroupSize*: The smallest number of entities in the group the mother spawns  
*maxGroupSize*: The largest number of entities in the group the mother spawns  
*repeats*: The number of times the mother should spawn children. The value *-1* means that the mother will spawn infinitely many groups. Any other value means that *that* many groups will be spawned, no more, no less  
*baseRepeatDelay*: The number of ticks between each time the mother spawns a group of children  
*searchRadius*: The search distance when looking for nearby mobs. This should be left constant  
*maxEntityCount*: The max number of entities in the search area. This should be left constant

#### Mount

When a Malevolent Mob is spawned, it may have another entity riding it. This entity can in theory be any other entity, but a living entity will work best. The rider may also be Malevolent, provided a role has been provided.

**Priority**: Low **Parameters**:  
*type*: The entity type that is to be the rider  
*role*: The role the ride should have as a Malevolent Mob. If the rider is not desired to be malevolent, the role can be set to *None* or an empty string

#### Poison Mist

The poison mist is an aura that will apply a poison status effect to any entity within it as long as the entity remains inside the aura. As soon as they move out, the effect will expire, unless they have another active poison effect that lasts for a longer amount of time. This aura is visualized as a cloud of brown particles around the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the aura, how far it can reach from the center of the mob  
*base/scaleStrength*: The strength of the poison effect entities inside the reach of the aura receives  
*affectPlayersOnly*: Indicates if only players can be damaged by the aura, or if all entities can be damaged by it

#### Potion Effect

Any Malevolent Mob can utilize the various potion effects that Minecraft has to offer. The effects can be said to last for a certain amount of time, with the potion effect reapplied every now and then. The strength and type of potion effect is completely customizable, where negative effects can also be applied using a negative strength.

Any potion effect type that is specified under

https://github.com/Bukkit/Bukkit/blob/master/src/main/java/org/bukkit/potion/PotionEffectType.java

is a valid potion effect type.

**Priority**: Normal **Parameters**:  
*effect*: The effect that that should be applied  
*base/scaleDuration*: The number of ticks the effect should last  
*base/scaleStrength*: The strength of the potion effect *base/scaleStartupDelay*: The number of ticks that must pass before the effect is first applied  
*base/scaleRepeatDelay*: The number of ticks between each successive reapplying of the effect  
*repeats*: The number of times the effect can be reapplied. If set to *-1*, the effect can be reapplied an infinite amount of times

#### Regeneration

There are almost no mobs that naturally regenerate health, yet that is in many situations something that one would expect them to be doing. As such, the regeneration passive exists, which allows mobs to regenerate health when they have been outside of combat for a certain amount of time. Each time the effect ticks, the mob will regenerate a given amount of health, unless they take any damage. By taking damage, the mob will be considered in combat, losing the regeneration ticking.

**Priority**: Normal **Parameters**:  
*base/scaleRegeneration*: The amount of health regenerated each tick  
base/scaleStartupDelay: The number of ticks for which the mob must not be attacked in order to start recharging the exoskeleton  
*base/scaleRepeatDelay*: The number of ticks between each successive recharge  
*repeat*: The number of times that the regeneration can tick before depleting. If set to *-1*, the regeneration can tick an infinite amount of times

#### Vampirism

The vampirism ability prevents any nearby players from being able to gain health from any instant health gain. In effect, this means that the only ways for a player to gain more health is by using a potion of health regeneration or natural regeneration, or move away from the mob and then use some ability that returns health or drink a healing potion. This aura is visualized as a cloud of red particles around the mob.

It is recommended that this ability is never used together with the *deteriorating aura* ability, as that would make it impossible for the player to regain health while in combat with the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the ability, how far it can reach from the center of the mob

#### Wither Aura

The wither aura is an aura that will apply a wither status effect to any entity within it as long as the entity remains inside the aura. As soon as they move out, the effect will expire, unless they have another active wither effect that lasts for a longer amount of time. This aura is visualized as a cloud of smoke particles around the mob.

**Priority**: Normal **Parameters**:  
*base/scaleRange*: The radius of the aura, how far it can reach from the center of the mob  
*base/scaleStrength*: The strength of the wither effect entities inside the reach of the aura receives  
*affectPlayersOnly*: Indicates if only players can be damaged by the aura, or if all entities can be damaged by it

#### Wither Particles

// Write me

## Active abilitives

// Everything about actives. Also mention the cooldown parameter which is global for all actives

# Roles

// How to set up a custom role and modify/use the default roles

The role of a Malevolent Mob is split up into several sections.

#### General

**Parameters:**   
*parentRole*: (Optional) Specifies the parent role; inherits all the settings from the specified role. Any settings defined in this role will overwrite the inherited settings.  
*customName*: (Optional) The custom name of the mob with this role; supports colors and styles with the ‘&’ symbol. This uses standard Minecraft color codes.  
*hideInfo*: (Optional) If true, the mob’s information won’t be displayed in the boss bar – only the name will be shown.  
*hideBossbar*: (Optional) If true, the mob’s boss will be disabled entirely, no information will be shown to the players.  
*level*: The desired level of the mob. Can be either a number (does not need to be an integer), or one of the following keys:  
 - *random*: Provides a completely random level from 1 to the max level  
 - *playerAverage*: Computes the average level of all nearby players and uses that. Defaults to 1 if no  
 players were found nearby  
 - *max:<scale>*: Takes the max level and multiplies with the number *scale*, that is the mob’s level.  
 *scale* is a number and must be separated from the tag *max* with the colon symbol, ‘:’  
*base/scaleHealth*: The base health of the mob, before being multiplied with the global health scale defined in the general section

#### Stats

Each mob will have some variation in their strengths and weaknesses against certain weapons and abilities. This section holds the information about the mob’s armor, and how quickly they are able to perform active abilities.

**Parameters:***base/scaleMeleeArmor*: The armor value against regular physical attacks  
*base/scaleMagicArmor*: The armor value against magical attacks  
*base/scaleArrowArmor*: The armor value against projectile attacks  
*base/scaleRangedArmor*: Additional armor value that is added if attacked from a distance  
*base/scaleAttackCooldown*: (Optional) How long the wait time is between each active ability being used, measured in seconds

#### Equipment

It is completely optional to include this section; if not included, the default behavior is to disable mobs picking up dropped items. The items carried by the mobs will be determined by Minecraft itself, and the drop rates will be untouched.

All parameters in this section are optional.

**Parameters:***allowPickup*: If set to true, the mob is able to pick up items dropped in the world  
*weaponItem*: The item the mob will be carrying; use *none* for no weapon   
*helmetItem*: The item the mob will be wearing as helmet; use *none* for no helmet  
*chestplateItem*: The item the mob will be wearing as chestplate; use *none* for no chestplate  
*leggingsItem*: The item the mob will be wearing as leggings; use *none* for no leggings  
*bootsItem*: The item the mob will be wearing as boots; use *none* for no boots  
*weaponDropChance*: The chance of the weapon being dropped, from 0 to 1  
*helmetDropChance*: The chance of the helmet being dropped, from 0 to 1  
*chestplateDropChance*: The chance of the chestplate being dropped, from 0 to 1  
*leggingsDropChance*: The chance of the leggings being dropped, from 0 to 1  
*bootsDropChance*: The chance of the boots being dropped, from 0 to 1

#### Passives / Actives

Passives:

Deteriorating\_Aura:

baseChance: -0.7

scaleChance: 0.03

Explosion: []

Actives:

Fire\_Burst: []

Ground\_Slam:

scaleStrength: 0.075

scaleKnockup: 0.003

Both passive and active abilities are provided to a mob in the same manner. The abilities can be provided by adding them to the list, and it’s possible to specify data for the abilities as well. If the default ability parameters is desired, appending the tag *[]* to the ability must be done.

Multiple effects of the same name can be added. In this case, it’s enough to add the tag *-<name>* to the end of the ability declaration. As an example, adding several potion effects can be done as

Example 3: How to add various abilities to a role, where the explosion and fire burst abilities have default parameters, while the other two have some custom parameters.

Potion\_Effect-someuniquename:  
 <parameters>  
Potion\_effect-anotheruniquename:  
 <parameters>

Any number of abilities can be added this way, provided the names are all unique.

#### Loot

Any mob can have custom loot, where the loot can vary significantly from mob to mob. It is possible to customize how much experience the mob grants, both for vanilla Minecraft as well as for SkillAPI, if it is installed. Mobs may also drop any number of items, which can have custom names and lore, custom enchantments and effects, and it is also possible to directly set the NBT data of the item.

**Parameters:***base/scaleVanillaExperience*: The maximum amount of vanilla experience gained from the mob  
*base/scaleSkillAPIExperience*: The maximum amount of SkillAPI experience gained from the mob *definitionFile*: The file that contains information about what items should be dropped as loot. Usually this will be the same name as the file containing the parameter *items  
base/scaleChance*: The chance of dropping the items  
*items*: A list of all the items that the mob can drop. Entries may be *'#random'* to indicate that the item should be generated from the random item generator.

Example 4: How to add three items to a mob, where two of the items are randomly generated and the third is a stack of 42 damaged swords with a red name.

items:  
 '0': '#random'  
 '1': '#random'  
 '2':  
 name: '&cMy Custom Item'  
 material: diamond\_sword  
 metadata: 314  
 amount: 42

# Item effects

Whenever a Malevolent Mob dies, by default they will drop an item that has some custom effect on it. These effects can be many different things, improving the item in various ways. Effects will have some drain on the tools, draining their durability when used, however.

What is common for all effects is that they can’t normally be applied to any item. Certain effects, such as a defensive effect like *Bulwark* can only be applied to armor. Effects like *Repair* can only be applied to items that can be damaged, and so on.

Certain effects can only be applied to a wand. A wand is a stick, a blaze rod or a bone. It’s highly recommended that no wand has more than one effect attached to it at any time, even though it is possible to do so. Certain wand effects will contradict each other, or be vastly different in how they function.

In terms on configuration, all effects may be disabled by changing the *enabled* flag in their configuration file to *false*. Item effects that have been disabled won’t be removed from items in play, but those effects will not be in effect when the item is used. These effects will not be attached to items that are dropped by Malevolent Mobs either, once the effect has been disabled. Do note that if an item is defined to contain a specific item effect which has been disabled, that item will not be dropped, and a different random item might be selected instead.

#### Armor Shredder

When applied to anything, whenever the player attacks a Malevolent Mob, their armor will be partially shredded and penetrated. The effect can shred a percentage and penetrate a flat amount using positive values, negative values enhances the armor rating while attacking with the effect.

**Main item group**: Weapons

**Parameters**:  
Armor shredded (percentage)  
Armor points penetrated

**Configurations**:  
*durabilityCostPerPercent*: The amount of durability lost per percent of armor shredded  
*durabilityCostPerPoint*: The amount of durability lost per point of armor penetrated  
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Bulwark

When applied to a piece of armor, the bulwark effect will modify physical damage by a certain amount. Physical attacks are melee attacks and projectiles, in addition to contact damage. When using positive values, the damage is decreased, negative values increases damage.

**Main item group**: Armor

**Parameters**:  
Damage modifier (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Charged

Attacking groups of mobs is often a tedious process, where only one target at a time can be attack. The charged effect will split the damage done to one target, evenly over a number of other nearby targets.

**Main item group**: Weapons

**Parameters**:  
Damage increase (percentage)  
Number of additional targets  
Spread range (decimeters)

**Configurations**:  
*durabilityCostPerDamage*: The durability cost per damage point dealt by the effect   
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Duplicator

When building, it is often inconvenient to have to go back to grab one more piece of block. This effect helps alleviate that issue, by making it possible to generate a new block from another one, at the cost of some charge.

**Main item group**: Wands

**Parameters**:  
Charge amount (tenth of a charge)  
Maximum charge (tenth of a charge)  
Amount recharged each minute (tenth of a charge)

**Configurations**:  
*blockValues*: The various values for certain blocks. Any block that is specified in this list can be duplicated, where the format is *<itemname>=<metadata>:<value>*, where the tag *=<metadata>* may be omitted

#### Earthmover

In order to remove vast quantities of earth and soil, much time is needed. With this effect, the amount of time needed is drastically reduced, as it allows removing blocks in a three by three square around the clicked block.

**Main item group**: Shovels

**Configurations**:  
*durabilityCostPerBlock*: The amount of damage the item takes for each harvested block

#### Entropy Limit

When applied to a piece of armor, this effect will modify wither damage by a certain amount. When using positive values, the damage is decreased, negative values increases damage.

**Main item group**: Armor

**Parameters**:  
Damage modifier (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Explosive Arrows

When using a bow, arrows aren’t always interesting. With this effect, any arrow that hits a living entity will explode, dealing additional damage to everything within the range of the explosion.

**Main item group**: Bows

**Parameters**:  
Explosion strength (twentieths of a heart)  
Explosion radius (decimeters)

**Configurations**:  
*durabilityCostPerUse*: The amount of damage the item takes for each exploding arrow

#### Fireward

When applied to a piece of armor, this effect will modify fire damage by a certain amount. Fire damage is damage due to lava and fire. When using positive values, the damage is decreased, negative values increases damage.

**Main item group**: Armor

**Parameters**:  
Damage modifier (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Fireworks

When applied to a wand, this item effect allows the player to launch fireworks around. The fireworks follow an arch and deals explosive damage where they land. Each launch consumes some charge from the wand.

**Main item group**: Wands

**Parameters**:  
Charge amount (tenth of a charge)  
Maximum charge (tenth of a charge)  
Amount recharged each minute (tenth of a charge)  
Small rocket damage (twentieths of a heart)  
Small rocket cooldown (ticks)  
Large rocket damage (twentieths of a heart)  
Large rocket cooldown (ticks)

**Configurations**:  
*costPerRocket*: The charge cost per launch of that rocket type   
*radius*: The radius of the explosion of that rocket type  
*velocity*: The launch velocity of that rocket type  
*rocketPower*: The flight power of that rocket type [Deprecated, no longer used]

#### Fragile

Any item costs some durability to use, but an item with this effect will lose some durability every time the item is used, either by breaking blocks, attacking, blocking or just clicking something.

**Main item group**: Items with durability

**Parameters**:  
Durability cost every use (hundredths of a point)

#### Frost

When the targets keep running away, it’s good to be able to slow them down. The frost effect will apply a slowing effect on the hit target for some duration, and optionally can apply some additional damage as well.

**Main item group**: Weapons

**Parameters**:  
Slow effect strength (percent) [Will be rounded to the nearest multiple of 15 internally]  
Slow duration (ticks)  
Additional damage (twentieths of a heart)

**Configurations**:  
*durabilityCostPerUse*: The durability cost per use of the frost effect   
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Growth

Whenever crops, trees, tall grass, sugar cane, cactus or even flowers aren’t growing fast, the growth effect can be used to accelerate the growth.

**Main item group**: Hoes

**Configurations**:  
*durabilityCostPerCrop*: The durability cost for each crop that was grown   
*durabilityCostPerTree*: The durability cost for each tree that was grown  
*durabilityCostPerGrass*: The durability cost for each tall grass that was grown  
*durabilityCostPerFlower*: The durability cost for each flower that was grown  
*durabilityCostPerCactus*: The durability cost for each cactus or sugar cane that was grown

#### Harvest

In order to speed up the tilling of soil, or harvesting of crops, the harvest ability can be utilized. When tilling soil, only dirt that is exposed to air will be tilled. When harvesting crops, only mature crops will be harvested; any harvested crop will also be replanted.

**Main item group**: Hoes

**Configurations**:  
*durabilityCostPerHarvest*: The durability cost for each crop that was harvested   
*durabilityCostPerTill*: The durability cost for each block that was tilled

#### Heated

Sometimes a furnace isn’t quick enough; this is where the heated effect can be used. Blocks harvested with this effect will be auto-smelted, if the block drops could be smelted in an otherwise mundane furnace. It does stack with the Fortune enchantment, meaning that it is possible to get multiple ingots from iron and gold ore.

**Main item group**: Shovels, pickaxes, axes

**Configurations**:  
*durabilityCostPerUse*: The durability cost for each drop that was smelted

#### Hungry

Whenever the player is in need of food, they might find themselves without any due to their hungry items. Every minute, this effect has a chance of consuming a unit of food from the inventory of the player.

**Main item group**: Everything

**Parameters**:  
Chance of eating food (percentage)

#### Lacerate

Certain targets have a huge pool of health and as such are really difficult to kill efficiently. The lacerate effect helps alleviate that issue, by dealing more damage the more health a target has.

**Main item group**: Weapons

**Parameters**:  
Additional damage from max health (tenths of percent)  
Additional damage from current health (tents of percent)

**Configurations**:  
*durabilityCostPerDamage*: The durability cost per time damage dealt   
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Levity

In order to make bows a more attractive weapon, this effect aims to improve bow damage as well as sniping potentials. It is possible to use this effect to fire an arrow that is unaffected by gravity, and to be sped up beyond what is normally possible. Negative values for the speed will slow down the arrow instead, reducing its damage.

**Main item group**: Bows

**Parameters**:  
Arrows ignore gravity (boolean)  
Arrow speed increase (percentage)

**Configurations**:  
*durabilityCostPerUse*: The amount of damage the item takes for each arrow that defies gravity  
*durabilityCostPerPercent*: The amount of damage the item takes for each percent the arrow is sped up

#### Magic Mirror

Many different attacks, effects and abilities do magic damage. This effect will reflect some of that damage back to the attacker, but not reduce damage dealt to the one wearing the effect.

**Main item group**: Armor

**Parameters**:  
Reflected damage (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each reflected point of damage

#### Magic Shield

When applied to a piece of armor, this effect will modify magic damage by a certain amount. Magic damage is potions, SkillAPI skills (if defined to be magic) and many various abilities from Malevolent Mobs and other item effects. When using positive values, the damage is decreased, negative values increases damage.

**Main item group**: Armor

**Parameters**:  
Damage modifier (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Modifier

WIP – will allow some altering of various item effects on items

#### Poison

Sometimes it is beneficial to do damage over time to targets. The poison effect will apply the poison status effect to any target that is attacked for some duration.

**Main item group**: Weapons

**Parameters**:  
Poison strength (potion level)  
Poison duration (ticks)

**Configurations**:  
*durabilityCostPerUsePerLevel*: The durability cost per level of the wither effect  
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Powerline

When applied to a wand, this item effect allows the player to attack with electricity. The player may use two modes, one a short-ranged bolt attack, and the other a long-range lightning bolt attack. Each use consumes some charge from the wand.

**Main item group**: Wands

**Parameters**:  
Charge amount (tenth of a charge)  
Maximum charge (tenth of a charge)  
Amount recharged each minute (tenth of a charge)  
Charge cost per lightning bolt (tenth of a charge)  
Charge cost per lightning strike (tenth of a charge)  
Lightning strike radius (decimeters)

**Configurations**:  
*lightningBoltDistanceCheck*: The distance checked for an intersection with any entity or block  
*lightningBoltCooldown*: The amount of time that must pass before the wand can be used again  
*lightningStrikeDistanceCheck*: The distance checked for an intersection with any entity or block  
*lightningStrikeCooldown*: The amount of time that must pass before the wand can be used again

#### Relic

For the lucky soul, sometimes a relic chest can appear. This chest contains one or more relics, which can be better items, or just more of the same level. They may have various negative effects, as well.

**Main item group**: Chests

**Parameters:**Number of items, plus chance of getting another one (percentage)  
Group of items (item group, string)  
Chance of negative effect (percentage)  
Group of negative effects (item effect group, string)

#### Repair

Items that are damaged can get annoying to repair as time progresses. This effect will repair the item over time, restoring the item to its full glory eventually. Alternatively, negative repair values will damage the item over time.

**Main item group**: Items with durability

**Parameters**:  
Repair amount per minute (points)

#### Runic Shield

In many situations, players need additional protection to save them from a sticky situation. The runic shield applies a few extra hearts of protection that must be depleted before the player’s health will be consumed. Certain effects such as starvation, poison, wither, drowning, falling and similar damage types won’t be blocked by the shielding.

**Main item group**: Armor

**Parameters**:  
Charge amount (tenth of a charge)  
Maximum charge (tenth of a charge)  
Amount recharged each minute (tenth of a charge)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Saturation

Eating all the time is annoying to have to deal with. This effect will restore some hunger from the player, as well as some saturation, every minute. Alternatively, using negative numbers, hunger and saturation will be consumed instead each minute.

**Main item group**: Armor

**Parameters**:  
Hunger restored (half shanks)  
Saturation restored (twentieths of a shank)

**Configurations**:  
*durabilityCostPerPoint*: The amount of damage the item takes for each hunger and saturation point that was restored

#### Sharpness

The sharpness effect allows weapons to do more damage to the targets. It can amplify the damage dealt to Malevolent Mobs, or all targets, in two ways. It can increase the damage by a percentage and/or a flat amount to either group. By using negative values, the damage dealt is decreased instead.

**Main item group**: Swords

**Parameters**:  
Malevolent Mob percentage damage increase (percentage)  
Malevolent Mob flat damage increase (half-hearts)  
All targets percentage damage increase (percentage)  
All targets flat damage increase (half-hearts)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each point of damage dealt

#### Shock Absorber

When applied to a piece of armor, this effect will modify explosion and electricity damage by a certain amount. When using positive values, the damage is decreased, negative values increases damage.

**Main item group**: Armor

**Parameters**:  
Damage modifier (percentage)

**Configurations**:  
*durabilityCostPerDamage*: The amount of damage the item takes for each blocked point of damage

#### Shroud

Taking vision away from the targets is a good way to ensure a getaway or get the upper hand in a fight. The shroud effect will effectively blind the target, applying the blindness status effect.

**Main item group**: Weapons

**Parameters**:  
Duration (ticks)

**Configurations**:  
*durabilityCostPerUse*: The durability cost per time the effect is applied   
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

#### Thaumic Bolt

When applied to a wand, this item effect allows the player to attack with magic bolts. Each use consumes some charge from the wand.

**Main item group**: Wands

**Parameters**:  
Charge amount (tenth of a charge)  
Maximum charge (tenth of a charge)  
Amount recharged each minute (tenth of a charge)  
Damage per bolt (twentieths of a heart)  
Bolt speed (decimeters per tick)

**Configurations**:  
*cooldownTime*: The number of ticks that needs to pass before the wand can be used again  
*damageIncrease*: The bolt damage increase per tick while in flight

#### Timber

Chopping down trees is a tedious exercise that gets boring rather quickly. The timber effect allow multiple logs to be chopped down in one go, allowing faster deforestation.

**Main item group**: Axes

**Parameters**:  
Number of blocks to harvest

**Configurations**:  
*durabilityCostPerBlock*: The durability cost per block harvested using this effect

#### Treasure

Sometimes it’s rather boring to go look for various metals and precious resources underground. The treasure effect can yield a chest of goodies that can be many different things to help the player progress.

**Main item group**: Chests

**Parameters**:  
Units of treasure (tenths of a unit)

**Configurations**:  
*itemValues*: The various values for certain items. Any item that is specified in this list can appear as treasure, where the format is *<itemname>=<metadata>:<value>*, where the tag *=<metadata>* may be omitted. The more often an item appears in the list, the more likely it is to be chosen as a treasure candidate

#### Tunneler

When digging, it takes a long time to progress through a bunch of rock. As such, it is handy to be able to quickly mine through a section of rock, which is where this effect comes into play. By using the effect, all stones in a three by three cube will be mined out.

**Main item group**: Pickaxes

**Configurations**:  
*durabilityCostPerBlock*: The durability cost per block harvested using this effect

#### Wither

Sometimes it is beneficial to do damage over time to targets. The wither effect will apply the wither status effect to any target that is attacked for some duration.

**Main item group**: Weapons

**Parameters**:  
Wither strength (potion level)  
Wither duration (ticks)

**Configurations**:  
*durabilityCostPerUsePerLevel*: The durability cost per level of the wither effect  
*bowDurabilityCostMultiplier*: The durability loss multiplier, if the weapon is a bow

# Loot and rewards

// More about how to configure the loot dropped by the mobs, such as defining new possible item effects, custom names and loot levels

# Permissions

There is only one permission available in the plugin:

malevolentmob.mmob

This permission gives access to the command /mMob, which in turn allows one to spawn Malevolent Mobs, modify existing roles, create and modify dungeons and everything else the plugin offers.

# Commands

All the commands used within the plugin are documented, where this documentation can be accessed from the command */mMob Help*. From here, it is possible to read everything about how the commands work, their syntaxes and what they do. Each command is documented within the plugin itself, through the */mMob Help <Instruction>* command.

# Debugging

// How to read the console or tell if something went wrong