WorldGuard Documentation Documentation

Release 7.0

WorldGuard Team

May 29, 2023

Contents

1 Documentation **3**

1.1 Installation . . . 3
1.1.1 Requirements . . . 3
1.1.2 Installation . . . 3
1.2 Configuration . . . 3
1.2.1 Configuration Files . . . 4
1.2.2 Settings . . . 4
1.2.3 Subpages . . . 15
1.3 Permissions . . . 16
1.3.1 Moderator . . . 16
1.3.2 Commands . . . 16
1.3.3 Regions . . . 17
1.3.4 Subpages . . . 19
1.4 Commands . . . 20
1.4.1 Gameplay . . . 20
1.4.2 Regions . . . 21
1.4.3 Emergency . . . 21
1.4.4 Troubleshooting . . . 22
1.4.5 Miscellaneous . . . 23
1.5 Blacklist . . . 23
1.5.1 Blacklist Files . . . 23
1.5.2 The Format . . . 24
1.5.3 Events . . . 24
1.5.4 Options . . . 25
1.5.5 Whitelist Mode . . . 25
1.5.6 Examples . . . 25
1.5.7 Logging . . . 26
1.6 Regions . . . 27
1.6.1 Contents . . . 27
1.7 Chest Protection . . . 62
1.7.1 Getting Started . . . 62
1.8 WorldGuard API . . . 62
1.8.1 As a Dependency . . . 62
1.8.2 Working with Regions . . . 65
1.8.3 From Bukkit Objects . . . 79
1.8.4 Internal APIs . . . 80

1.9 Advanced Topics . . . 80
1.9.1 Event Logging . . . 80
1.10 Common Questions . . . 83
1.10.1 General . . . 84
1.10.2 Can't Build . . . 84
1.10.3 Building Not Blocked . . . 85
1.10.4 Region Protection . . . 86
1.11 Getting Help . . . 86
1.12 Source Code . . . 86

2 Links **87**

3 Indices and Tables **89**

WorldGuard has a host of functions for server owners, server map makers, regular survival servers, and everyone else in between.

- *Create zones in your world* that only permit mods or certain players from building within
- Set *additional game rules* on your server (deny wither block damage, falling damage, etc.)
- Set *game rules on specific areas* (food regen, health regen, disable PvP, TNT, control mob damage) in your world
- *Blacklist certain items* and blocks so they can't be used
- Dump useful statistics and information about your server (/wg report -p)
- CPU profile your server (/wg profile -p)
- Add *useful commands* like an immediate "STOP ALL FIRE SPREAD" command.
- Works, more or less, *with other plugins and mods* (other Bukkit plugins and FTB mods)
- Protects *against many types of abuse* (tree growth, TNT cannons, piston machines, etc.)
- Certain interactions can also be allowed (door and lever usage, etc.)
- Open source, and one of the oldest Minecraft projects (older than Bukkit!)
- Enable only features you want! **Everything is off by default.** You can just install WorldGuard and configure it later.

CHAPTER 1

Documentation

1.1 Installation

1.1.1 Requirements

WorldGuard requires a version of the Minecraft server that supports the Bukkit API, which includes CraftBukkit, Spigot, and Paper. The "vanilla" Minecraft server cannot run plugins. Other server implementations, such as Forge or Sponge, are also not supported by WorldGuard.

The other requirement is the WorldEdit Bukkit plugin, a very lightweight in-game map editor, also made by us. Please note that you need the Bukkit version of WorldEdit, not another implementation.

1.1.2 Installation

WorldGuard can be downloaded from the Bukkit dev site.

1. In your server folder, create a "plugins" folder if one does not yet exist.
2. Copy the WorldGuard .jar file into the plugins folder. (Make sure you get WorldEdit as well!)
3. Start your server.

Check your server log for errors. If you encounter errors, see the *FAQ* page.

1.2 Configuration

Many of WorldGuard's features exist as configuration options (potion blocking, scuba mode with pumpkins, etc.). Every configuration option available is listed on this page.

1.2.1 Configuration Files

Once you have run your server with WorldGuard installed, you will find the main configuration file inside the **plugins/WorldGuard** folder:

- `config.yml`

Then for every world, you will find per-world configuration files:

- `worlds/world/config.yml`
- `worlds/world_nether/config.yml`
- `worlds/mining_world/config.yml`

If you open up the per-world configuration files, they will be nearly empty. When you wish to override a setting, you would copy it into the world's configuration file.

Example: Making a configuration option per-world

In the main configuration file, you may have set `block-creeper-block-damage` to true:

```
mobs:
    block-creeper-explosions: false
    block-creeper-block-damage: true
    block-wither-explosions: false
```

But you want to set it to `false` in your nether world. Open up `worlds/world_nether/config.yml` and replace the file with:

```
mobs:
    block-creeper-block-damage: false
```

Only the relevant line, as well as any parent sections, needs to be copied over.

1.2.2 Settings

Note: These options are presented here as a reference, but you should change a desired option by first finding it in `config.yml` because some settings may need to be nested under another setting.

Setting	Default	Description
op-permissions	TRUE	Whether players with op should be given all permissions for WorldGuard, even if the permissions plugin in use does not provide permission for ops.
summary-on-start	TRUE	Show summary information about WG's settings for each world on server start. This is pretty noisy and it should be disabled if you have many worlds.
auto-invincible	FALSE	Give players with the `worldguard.auto-invincible` permission invincibility mode on join.
auto-invincible-group	FALSE	Give players in the `wg-invincible` permission group invincibility mode on join.
auto-no-drowning-group	FALSE	Give players in the `wg-amphibious` permission group water breathing mode on join.
use-player-move-event	TRUE	Whether WorldGuard should use (a little) more CPU to handle features that require tracking player movement. This must be on to use healing, feeding, greeting, and some other *Region Flags*.
use-player-teleports	TRUE	Whether teleport events should be considered when tracking player movement. This should always be on if you are using the player move event and any of the flags that require that.
host-keys		A list of hostnames that players must connect from. See *Host Keys*.

security.*

Setting	Default	Description
deop-everyone-on-join	FALSE	Clear op status from all players that join.
block-in-game-op-command	FALSE	Block the /op command from being used in-game.

build-permission-nodes.*

Setting	Default	Description
enable	FALSE	A feature that lets you block building based on giving players the proper permissions. See *Build Permissions*.
deny-message		Concerning build permissions, this is the message that is sent when permission is denied. If a message is not set, a default one is used.

event-handling.*

Setting	Default	Description
block-entity-spawns-with-untraceable-cause	FALSE	As Bukkit does not always tell plugins the exact reason that an entity was spawned, it may be possible for a player to bypass protection to spawn an entity (such as with a spawn egg). This option blocks cases where the true cause cannot be determined. It is recommended that this option is left off because the number of cases where the cause is not known is quite large.
interaction-whitelist	[]	A list of block types that should not be protected. For example, if chests were added to this list, then they would never be protected with *region protection*. This setting is useful primarily when non-vanilla functionality is present (game features added by other plugins or mods) and you don't want it blocked.
emit-block-use-at-feet	[]	A list of items that, if used, will also require that the player have the permission to *theoretically* modify the block at his or her feet. This setting is useful primarily when there is some item from some plugin or mod that uses a projectile (that affects the world) but does not test permission with WorldGuard. However, this is not a proper solution because the player can still stand in an area where he or she has permission and shoot *into* the desired area.
ignore-hopper-item-move-events	FALSE	Disable protections relating to hoppers moving items to and from containers. This can be set to true to slightly improve performance if your server has many hoppers, but be warned that it will allow hoppers outside regions to pull items from inside regions if they are close enough to the border.
break-hoppers-on-denied-move	TRUE	If a hopper attempts to pull an item and is denied (as long as the previous config setting is false), WorldGuard will break the hopper block to prevent it from continuously attempting to pull items.

Example: Disabling protection on workbenches

The `interaction-whitelist` option can be used:

```
interaction-whitelist: [workbench]
```

protection.*

Setting	Default	Description
item-durability	TRUE	Set to `false` to have items never break.
remove-infinite-stacks	FALSE	Remove items with 'infinite' stack sizes, which is essentially any stack size less than 0 (which is the result of a number overflowing the maximum and wrapping into the negatives).
disable-xp-orb-drops	FALSE	Whether to disable XP orb drops.
use-max-priority-association	FALSE	Wheter or not non-player associables, such as pistons, are only members of the regions with the highest priorities in which they are in. If set to `false`, non-player associables are members of all regions in which they are in, which means that pistons, for example, can push blocks from inside a region outwards into a surrounding region (unless there is only a protected *Global Region* outside). If set to `true`, pistons, for example, cannot push blocks from inside a region outwards into a surrounding protected region with a lower priority.

gameplay.*

Setting	Default	Description
block-potions	[]	A list of potion types that cannot be used. The list of possible potion types can be found in Bukkit.
block-potions-overly-reliably	FALSE	Whether WorldGuard should try extra hard to block the list of potions mentioned in *block-potions*. This is generally not needed and enabling this may block more than you want.

Example: Blocking the use of night vision and speed potions

The names found in Bukkit are used:

```
block-potions: [night_vision, speed]
```

simulation.sponge.*

Setting	Default	Description
enable	FALSE	Whether to simulate sponge blocks similar to the way they worked in Minecraft Classic. Between the introduction of survival Minecraft and Minecraft 1.8 (several years), sponges did not work, but this setting is now obsolete and its use is no longer recommended.
radius	3	The radius of the sponge's action.
redstone	FALSE	Whether Redstone can control the simulated Sponge blocks.

default.*

Setting	Default	Description
pumpkin-scuba	FALSE	Whether players with pumpkins (but not Jack o' Lanterns) in their helmet slot will have water breathing.
disable-health-regain	FALSE	Whether automatic health regeneration should be disabled.

physics.*

Setting	Default	Description
no-physics-gravel	FALSE	Whether gravel should not fall.
no-physics-sand	FALSE	Whether sand should not fall.
vine-like-rope-ladders	FALSE	Whether ladders will work like vines in that they won't break as long as the top most ladder block is in a valid location (it's on a wall).
allow-portal-anywhere	FALSE	Whether portal blocks can be placed in invalid locations.
disable-water-damage-blocks	[]	A list of block types that will not be broken by water.

Example: Preventing Redstone and Redstone torches from being damanged by water

Material names are from Bukkit's Material list:

```
physics:
    disable-water-damage-blocks: [redstone_wire, redstone_torch]
```

ignition.*

Setting	Default	Description
block-tnt	FALSE	Whether the detonation of TNT should be blocked.
block-tnt-block-damage	FALSE	Whether TNT should do no block damage.
block-lighter	FALSE	Whether the use of flint and steel should be disabled.

fire.*

Setting	Default	Description
disable-lava-fire-spread	FALSE	Whether lava should be able to create fires.
disable-all-fire-spread	FALSE	Whether fire can spread.
disable-fire-spread-blocks	[]	A list of block types that fire cannot spread to, or at least damage.
lava-spread-blocks	[]	If set (as a list of block types), the only blocks on which lava could flow on (other than air) would be the ones in the list.

mobs.*

Setting	Default	Description
block-creeper-explosions	FALSE	Whether the effects of Creeper explosions should be disabled.
block-creeper-block-damage	FALSE	Whether block damage caused by Creeper explosions should be disabled.
block-wither-explosions	FALSE	Whether the effects of Wither explosions should be disabled.
block-wither-block-damage	FALSE	Whether block damage caused by Wither explosions should be disabled.
block-wither-skull-explosions	FALSE	Whether the effects of Wither skulls should be disabled.
block-wither-skull-block-damage	FALSE	Whether block damage caused by Wither skulls should be disabled.
block-enderdragon-block-damage	FALSE	Whether block damage caused by Enderdragons should be disabled.
block-enderdragon-portal-creation	FALSE	Whether the ability of the Enderdragon to create a portal should be disabed.
block-fireball-explosions	FALSE	Whether the effects of fireball explosions should be disabled.
block-fireball-block-damage	FALSE	Whether block damage caused by fireball block damage should be disabled.
anti-wolf-dumbness	FALSE	Whether the wolf should be invincible in a number of situations, including, but not limited to, walking into lava and getting stuck. When wolves were first introduced into the game, Minecraft had very poor path finding and so wolves would frequently walk into lava or fire. However, as the AI of helper mobs in Minecraft have still much to improve, this setting may still prove to be useful.
allow-tamed-spawns	TRUE	Whether tamable mobs (wolves, horses, cats, etc.) should be spawnable.
disable-enderman-griefing	FALSE	Whether the ability of Endermen to pick up and place blocks should be disabled.
disable-snowman-trails	FALSE	Whether the feature of snowmen placing snow trails should be disabled.
block-painting-destroy	FALSE	Whether the ability of mobs to break paintings should be disabled.
block-item-frame-destroy	FALSE	Whether the ability of mobs to break item frames should be disabled.
block-armor-stand-destroy	FALSE	Whether the ability of mobs to break armor stands should be disabled.
block-plugin-spawning	TRUE	Whether mobs spawned by plugins should be blocked when needed to apply some of these configuration options or to protect areas of the world.
block-above-ground-slimes	FALSE	Whether slimes spawning above ground should be disabled.
block-other-explosions	FALSE	Whether miscellaneous explosions should be disabled.
block-zombie-door-destruction	FALSE	Whether the ability for zombies to break doors should be disabled.
block-creature-spawn	[]	A list of entity types that should not spawn.

player-damage.*

Setting	Default	Description
disable-fall-damage	FALSE	Whether fall damage should be disabled for players.
disable-lava-damage	FALSE	Whether lava damage should be disabled for players.
disable-fire-damage	FALSE	Whether fire damage should be disabled for players.
disable-lightning-damage	FALSE	Whether lightning damage should be disabled for players.
disable-drowning-damage	FALSE	Whether drowning damage should be disabled for players.
disable-suffocation-damage	FALSE	Whether suffocation damage should be disabled for players.
disable-contact-damage	FALSE	Whether contact damage (cacti, etc.) should be disabled for players.
teleport-on-suffocation	FALSE	Whether players should be teleported to a safe location (if found, and which is usually up) when they are suffocating. This feature potentially allows players to cross vertical barriers.
disable-void-damage	FALSE	Whether void damage (falling into the bottom of the world) should be disabled for players.
teleport-on-void-falling	FALSE	Whether players should be teleported to a safe location (if found) when they are falling into the void. This feature potentially allows players to enter areas that they normally may not be able to go.
reset-fall-on-void-teleport	FALSE	Resets fall distance on void teleporting. Leaving this off will likely let the player die of fall damage (though their items will not be in the void, at least).
disable-explosion-damage	FALSE	Whether explosion damage should be disabled for players.
disable-mob-damage	FALSE	Whether damage from mobs should be disabled for players.
disable-death-messages	FALSE	Whether death messages for players should be disabled.

crops.*

Setting	Default	Description
disable-creature-trampling	FALSE	Disable the trampling of farmland by creatures.
disable-player-trampling	FALSE	Disable the trampling of farmland by players.

turtle-egg.*

Same options as the crops section, but for turtle eggs.

weather.*

Setting	Default	Description
prevent-lightning-strike-blocks	[]	A list of block types where lightning should not be able to strike.
disable-lightning-strike-fire	FALSE	Whether fire caused by lightning should be blocked.
disable-thunderstorm	FALSE	Whether thunderstorms should never start.
disable-weather	FALSE	Whether weather events (including thunderstorms) should never start.
disable-pig-zombification	FALSE	Whether the "zombification" of pigs when they are struck by lightning should be disabled.
disable-powered-creepers	FALSE	Whether the possibility of creepers becoming powered when they are struck by lightning should be disabled.
always-raining	FALSE	Whether it should always be raining or snowing.
always-thundering	FALSE	Whether it should always be thundering.

dynamics.*

Setting	Default	Description
disable-mushroom-spread	FALSE	Whether the spread of mushrooms should be disabled.
disable-ice-melting	FALSE	Whether the melting of ice should be disabled.
disable-snow-melting	FALSE	Whether the melting of snow should be disabled.
disable-snow-formation	FALSE	Whether the formation of snow should be disabled.
disable-ice-formation	FALSE	Whether the formation of ice should be disabled.
disable-leaf-decay	FALSE	Whether the decay of leaves should be disabled.
disable-grass-growth	FALSE	Whether the growth of grass should be disabled.
disable-mycelium-spread	FALSE	Whether the spread of mycelium should be disabled.
disable-vine-growth	FALSE	Whether the growth of vines and kelp should be disabled.
disable-rock-growth	FALSE	Whether the growth of rocks such as dripstones should be disabled.
disable-sculk-growth	FALSE	Whether the growth of sculk should be disabled.
disable-crop-growth	FALSE	Whether the growth of wheat, carrots, melons, etc should be disabled.
disable-soil-dehydration	FALSE	Whether the dehydration of soil should be disabled.
disable-coral-block-fade	FALSE	Whether coral should remain alive when not in water.
disable-copper-block-fade	FALSE	Whether the oxidation of copper blocks should be disabled.
snow-fall-blocks	[]	If set (as a list of block types), the only blocks on which snow can fall on would be the ones in the list.

chest-protection.*

Setting	Default	Description
enable	FALSE	Enables *Chest Protection*.
disable-off-check	FALSE	Even if chest protection is off, WorldGuard will block the creation of signs with `[Lock]` on them so that if chest protection is later enabled, players cannot have preemptively lock chests that they did not own to begin with. This option, if set to `true`, disables this check when chest protection is off.

blacklist.*

Setting	Default	Description
use-as-whitelist	FALSE	Whether the purpose of the blacklist should be inverted, where the only things that can be done are the ones that are denied on the blacklist.

blacklist.logging.*

These settings determine what the "log" action in the *Blacklist* does. Several logging targets can be enabled simultaneously.

Setting	Default	Description
console:		
enable	TRUE	Whether logging to the console should be enabled.
database:		
enable	FALSE	Whether logging to a database should be enabled.
dsn	jdbc:mysql://localhost:3306/minecraft	The connection string for the database. `minecraft` in the default is the name of the database.
user	root	The username to connect to the database with.
pass		The password to connect to the database with.
table	blacklist_events	The table to use.
file:		
enable	FALSE	Whether logging to files should be enabled.
path	worldguard/logs/%Y-%m-%d.log	The pattern for the log files.
open-files	10	The maximum number of file handles to have open at once (file handles with the operating system).

regions.*

Hint: You cannot override `use-creature-spawn-event` per-world.

Setting	Default	Description
wand	leather	The ID of the item that is used to right click a block to inspect the regions affecting it. By default, this item is cow leather. Before, it was string but Minecraft added a use for string.
disable-bypass-by-default	FALSE	Whether bypass permissions are disabled by default.
announce-bypass-status	FALSE	Whether a hint for `/region bypass` should be displayed at login.
invincibility-removes-mobs	FALSE	If enabled, if a player is attacked while he or she is invincible due to the invincibility *region flag*, then the attacking mob is removed from the world.
fake-player-build-override	TRUE	Whether players with a name that start with `[` and end with `]` should bypass all protection. This is only the case with third-party plugins and mods that use 'fake players' in lieu of firing the proper events.
explosion-flags-block-entity-damage	TRUE	Whether the various `-explosion` *region flags* should also disable entity damage when enabled.
high-frequency-flags	FALSE	Whether to handle frequently occurring events such as fire spread or fluid flow in regards to *region flags* (such as `fire-spread`, `water-flow`, and `lava-flow`). This is disabled by default because those mentioned flags are rarely used and this setting may result in a performance impact in certain scenarios (many, many players and many, many regions).
protect-against-liquid-flow	FALSE	Whether fluid flows between regions should be blocked. This prevents one player from griefing another by having lava or water flow into the other player's region. This setting must be enabled in tandem with `high-frequency-flags` for it to take effect.
use-paper-entity-origin	FALSE	When on a Paper server, this option will treat entities as members of the region where they spawned, not where they currently are. This will automatically prevent mobs that wander into regions from accidentally destroying it. (Note that this is separate from mobs that are targetting players in regions - that is still dependent on the player's permission, not the mob's.)
max-claim-volume	30000	The maximum number of blocks in a region that can be claimed with *self-serve region claiming*.
claim-only-inside-existing-regions	FALSE	Whether players can only claim within existing regions.
max-region-count-per-player:		The maximum number of regions that can be claimed by a player (via *self-serve region claiming*). This setting can differ per permission-group if new entries are added below (like for 'default'). 'default' is the default limit. If a player is a member of several groups that are listed, then the player receives the highest limit.
default	7	(See above.)

Example: Setting different region count limits per group

Each permission group is given its own entry:

```
max-region-count-per-player:
    default: 7
    builders: 20
    moderators: 40
```

Warning: There are also some additional settings in this category, but we do not recommend changing them:

Setting	Default	Description
enable	TRUE	Whether support for defining regions should be enabled.
use-creature-spawn-event	TRUE	Whether the creature move event should be handled for applying some mob-related *region flags*.

regions.uuid-migration.*

These settings are used to migrate from old versions of WorldGuard and Minecraft. In the past, Minecraft accounts were identified purely by their name, but around the end of Minecraft 1.7, Minecraft moved to a system where players were (internally) identified by "UUIDs" and players *could change their names*. However, all older configuration files still referred to players by their name, so the following settings enable the conversion to UUID on server start for one time (the setting is automatically disabled).

UUID migration can be run repeatedly (with repeated changing of the setting) and it will only convert names that have not yet been converted to UUIDs. If there are no names to convert, then nothing will happen.

Hint: You cannot override these settings per-world.

Setting	Default	Description
perform-on-next-start	FALSE	Whether UUID migration should be performed on next server start (once). If the configuration file does not yet have this setting, WorldGuard will assume its value is `true`, perform the conversion, then set the setting to `false`.
keep-names-that-lack-uuids	TRUE	Sometimes, a name that was added to a region does not actually exist. This option keeps names that can't be converted in the region data so that they can later be removed or re-converted (by adjusting the settings in this section to re-run the conversion).

regions.sql.*

Hint: You cannot override these settings per-world.

Setting	Default	Description
use	FALSE	Whether MySQL should be used to store data (see *Storage Drivers*).
dsn	jdbc:mysql://localhost/worldguard	The connection string for the database. `worldguard` in the default is the name of the database.
username	worldguard	The username to connect to the database with.
password	worldguard	The password to connect to the database with.
table-prefix		The table prefix.

1.2.3 Subpages

Host Keys

Frequently in the past, Minecraft had failures in its login code where players could login to a server as any player, including administrators and moderators. Between 2010 and 2013, exploits of this nature were made public five times, frequently leading to thousands of servers being hacked.

The host keys feature was added to WorldGuard as an extra barrier to impersonation. It works because an extra piece of information, not known by Mojang, has to be sent from the client during login to a server. Even if an attacker were able to break Minecraft's login system and join as a moderator, because the attacker's game would lack this piece of information, the server could detect impersonation.

Note: Security breaches of this nature are less common these days.

How It Works

When a player connects to a server with an address, say `play.example.com`, Minecraft will tell the server that the player connected with that address. A moderator could connect to a special, secret `secretmod.play.example.com` address, and the server could easily check whether the address used by the moderator started with `secretmod`.

The host keys feature allows you to configure an an accepted address for certain players. If a player on the list connects with an incorrect address, he or she is kicked immediately.

Configuration

Setup is done using the *Configuration*:

```
host-keys:
    your_username: bagels.play.example.com
    moderator1_name: manoverboard.play.example.com
```

Tip: Host keys support uuids, and it's recommended you assign players' uuids to host keys instead of usernames in case they change their username.

DNS Configuration

To make this work, you have to make `bagels.play.example.com` and `manoverboard.play.example.com` point to your server. However, you should **not** add specific records for the domains that you use, because this allows attackers to easily figure out the secret domains.

Rather, it is recommended that you setup "wildcard addresses." An example of a wildcard address may be `*.play.example.com`, which would mean that *any* prefix would work (`aa.play.example.com`, `ab.play.example.com`, `ac.play.example.com`, etc.).

Tip: If you don't have a domain name or can't set a wildcard address, you can use xip.io.

Alternatives

An alternative to host keys, although not provided by WorldGuard, is to use some sort of login command that takes a password.

1.3 Permissions

By default, no one can use WorldGuard. In order for yourself, moderators, and players to use WorldGuard, you must provide the proper permissions. One way is to provide op to moderators and administrators (unless disabled in the *configuration*), but providing the permission nodes on this page (through a permissions plugin) is the more flexible.

You can give the `worldguard.*` permission to give yourself and other administrators full access to WorldGuard.

1.3.1 Moderator

Permission	Explanation
worldguard.notify	Receive notification with the 'notify' *blacklist* action and the 'notify' *region flags*.
worldguard.region.bypass.<world>	Bypass region protection for a given world, except for PvP deny flags.
worldguard.region.toggle-bypass	Allows using the `/rg bypass` command to toggle the above behavior (still requires the bypass permission).
worldguard.chest-protection.override	Bypass *Chest Protection* and open protected chests.

Warning: If you are op or have all permissions, you will implicitly have these protection bypass permissions and it will appear that protection does not work.

1.3.2 Commands

See the *Commands* page for an explanation of some of these commands.

Permission	Explanation
worldguard.god	Be able to use `/god` (and `/ungod`).
worldguard.god.other	Be able to use `/god` (and `/ungod`) on others.
worldguard.heal	Be able to use `/heal`.
worldguard.heal.other	Be able to use `/heal` on others.
worldguard.slay	Be able to use `/slay`.
worldguard.slay.other	Be able to use `/slay` on others.
worldguard.locate	Be able to use `/locate`.
worldguard.stack	Be able to use `/stack`.
worldguard.stack.illegitimate	Be able to make stack sizes with `/stack` that exceed normal limits (i.e. 64 buckets in one stack).
worldguard.fire-toggle.stop	Be able to use `/stopfire` and `/allowfire`.
worldguard.halt-activity	Be able to use `/stoplag`.
worldguard.reload	Be able to use `/wg reload`.
worldguard.report	Be able to use `/wg report`.
worldguard.profile	Be able to use `/wg profile`.
worldguard.report.pastebin	Be able to have a report or profile submitted to a pastebin service.
worldguard.flushstates	Be able to use `/wg flushstates`.
worldguard.running	Be able to use `/wg running`.
worldguard.debug.event	Be able to use the event testing commands under `/wg debug`. Those commands are used to simulate an action so you can trace down, for example, which plugin is blocking player versus player combat. However, this essentially allows a moderator to impersonate another player.

1.3.3 Regions

Permission	Explanation
worldguard.region.wand	Be able to use the *Region Wand*.
worldguard.region.load	Be able to use `/rg load`.
worldguard.region.save	Be able to use `/rg save`.
worldguard.region.migratedb	Be able to use `/rg migratedb`.
worldguard.region.migrateuuid	Be able to use `/rg migrateuuid`.
worldguard.region.define	Be able to use `/rg define`.
worldguard.region.claim	Be able to use `/rg claim`.
worldguard.region.unlimited	Bypass claiming limits.
worldguard.region.list	Be able to use `/rg list`.
worldguard.region.list.own	Be able to use `/rg list` and have it show one own's regions.

Per-Region Commands

The following commands support a base permission (such as `worldguard.region.redefine`), but also allow you to use specific permissions that only apply if a player is an owner or member of a region (note that an owner is also a member):

- `worldguard.region.redefine.own.<region name>`
- `worldguard.region.redefine.member.<region name>`
- `worldguard.region.redefine.<region name>`

Permissions systems that support wildcard permissions can be used to grant permissions to all regions in any given category, e.g.:

- `worldguard.region.redefine.*`

Permission	Explanation
worldguard.region.redefine.*	Be able to use `/rg redefine`.
worldguard.region.remove.*	Be able to use `/rg remove`.
worldguard.region.setpriority.*	Be able to use `/rg setpriority`.
worldguard.region.setparent.*	Be able to use `/rg setparent`.
worldguard.region.select.*	Be able to use `/rg select`.
worldguard.region.info.*	Be able to use `/rg info` and `/rg flags`.
worldguard.region.teleport.*	Be able to use `/rg teleport`.
worldguard.region.addmember.*	Be able to use `/rg addmember`.
worldguard.region.addowner.*	Be able to use `/rg addowner`.
worldguard.region.removemember.*	Be able to use `/rg removemember`.
worldguard.region.removeowner.*	Be able to use `/rg removeowner`.

Example: Letting players look up information on only regions that they own

Use the own.* form of the permission:

```
worldguard.region.info.own.*
```

Example: Allowing every player to use `/rg teleport city`

The region name can be specified in the permission:

```
worldguard.region.teleport.city.*
```

Flag Command

The `/rg flag` command has the basic permission:

```
worldguard.region.flag.*
```

However, rather than providing that encompassing permission, you can provide the combination of the following two:

- To determine *which regions* can have their flag changed by the player, any of the following permissions can be given (and they work like the permissions in the previous section):
  - `worldguard.region.flag.regions.own.<region name>`
  - `worldguard.region.flag.regions.member.<region name>`
  - `worldguard.region.flag.regions.<region name>`
- However, to determine *which types of flags* can be set by the player, permissions of the following patterns can be given:
  - `worldguard.region.flag.flags.<flag name>.<flag value>.own.<region name>`

  - `worldguard.region.flag.flags.<flag name>.<flag value>.member.<region name>`
  - `worldguard.region.flag.flags.<flag name>.<flag value>.<region name>`

Examples for `<flag value>` are `allow`, `deny`, `unset` (if no value is specified), etc. More complex values are also checked, however, special characters such as `,`, `.`, whitespaces, etc. are omitted.

Example: Letting players be able to only change flags on regions that they own, and limit the flags that they can change to `use` and `chest-access`

You would need to provide the following permissions:

```
worldguard.region.flag.regions.own.*
worldguard.region.flag.flags.use.*
worldguard.region.flag.flags.chest-access.*
```

1.3.4 Subpages

Build Permissions

An optional feature lets you provide build permissions based on permission nodes. It can be enabled in the *Configuration*:

```
build-permission-nodes:
    enable: true
    deny-message:
```

An optional message can be set to change the message that players receive when an action is blocked.

Permissions List

- Block place: `worldguard.build.block.place.<material>`
- Block break: `worldguard.build.block.remove.<material>`
- Block interact: `worldguard.build.block.interact.<material>`
- Entity spawn: `worldguard.build.entity.place.<type>`
- Entity destroy: `worldguard.build.entity.remove.<type>`
- Entity interact: `worldguard.build.entity.interact.<type>`
- Entity damage: `worldguard.build.entity.damage.<type>`
- Item use: `worldguard.build.item.use.<material>`

The permissions are also checked in the style of `worldguard.build.block.<material>.<action>`, so `worldguard.build.block.<material>.place` would work too.

The list of usable material names comes from the Material list in Bukkit. For example, the permission for placing the bed block would be `worldguard.build.block.place.red_bed`. Be aware that *Material* contains both item and block names.

1.4 Commands

WorldGuard provides five categories of commands:

- Gameplay-related (`/god`, `/heal`, etc.)
- Working with *regions*
- Time-sensitive emergency response
- Troubleshooting
- WorldGuard-related (version information, reloading configuration)

See the *permissions page* for the necessary permission nodes.

Hint: When a parameter (like `<player>`) is surrounded by `[]`, it means that the parameter is optional. In addition, don't put < or > when entering the command.

1.4.1 Gameplay

Command	Parameters	Explanation
/god	[-s] [<player>]	Give yourself or another player invicinbility. `-s` can be optionally specified to have no message emitted.
/ungod	[-s] [<player>]	Remove invicinbility from yourself or another player. `-s` can be optionally specified to have no message emitted.
/heal	[-s] [<player>]	Heal yourself or another player. `-s` can be optionally specified to have no message emitted.
/slay	[-s] [<player>]	Slay yourself or another player. `-s` can be optionally specified to have no message emitted.
/locate	[<player>]	Have your compass point to another player (if specified), or pointed to spawn if no player is specified.
/stack		Organize your inventory and automatically stack items.
/;		An alias for `/stack`.

Selectors

The commands that take a player name support *selectors*, but they differ from Minecraft's selectors which came about 3 years later.

Players can be matched with:

- By default, players are matched if their name *starts with* the given name
- `*` to match all
- `#<world>` to match everyone on a certain world
- `#near` to match nearby players to the command's user
- `@<name>` to match a player's name exactly

1.4.2 Regions

Command	Parameters	Explanation
/wg flushstates [player]		Debugging command that clears information about players that is stored for the purpose of applying *region flags*. For example, for the `entry` flag to work, the player's last 'block' is stored every time he or she moves, which is used to determine whether the player has entered a region. This command can be useful when a player is stuck inside or outside a region due to the `entry` or `exit` flags, especially after changes in membership, permission groups, etc. The optional player argument will reset states only for the provided player; if not supplied, all players' flag states will be reset.

Additional region commands can be found in the *regions section*.

1.4.3 Emergency

Command	Parameters	Explanation
/stopfire	[<world>]	Immediately stop all fire spread on the current or given world.
/allowfire	[<world>]	Lift the fire spread suspension.
/stoplag	[-s]	Remove all entities in all worlds and enter a mode where all physics events and liquid flow events are stopped. In addition, when a chunk loads, all entities in that chunk are removed. The purpose of this command is to solve problems involving thousands of entities or items. However, when this command was added, tameable animals were not a thing and the loss of entities was not an issue, but this is no longer the case and this command should be reserved for extremely desperate occasions. `-s` can be provided to make the toggle not issue an announcement in chat.
/stoplag	-c [-s]	Disable the 'stop lag' mode. `-s` can be provided to make the toggle not issue an announcement in chat.
/stoplag	-i	Display the status of the 'stop lag' mode.

1.4.4 Troubleshooting

Command	Parameters	Explanation
/wg report	[-p]	Writes a report file (placed at `plugins/WorldGuard/report.txt`) detailing information about the server setup (list of plugins, their versions, world settings) as well as WorldGuard's configuration. This command is to easily provide information to others for support purposes. The report does not contain sensitive data. Use `-p` to also submit the report to a pastebin site and have a link generated that you can give to others.
/wg profile	[-p] [-t <name>] [<minutes>]	Starts profiling CPU usage of the current running server, and runs the profiler for the given duration (defaulting to 5 minutes if a duration is not specified). The profiler is based on WarmRoast. By default, results are only collected for the main thread where the world is 'ticked', but `-t` can be specified to filter by a different thread name (case in-sensitive). An asterisk (`*`) can be specified instead to profile all threads (i.e. `-t *`). Use `-p` to also submit the profiling result to a pastebin site and have a link generated that you can give to others. This is recommended because the output of the profiler is formatted by the pastebin site to make it readable.
/wg debug testbreak	[-t] <player>	Simulates a 'block break' event. This is explained below.
/wg debug testplace	[-t] <player>	Simulates a 'block place' event. This is explained below.
/wg debug testinteract	[-t] <player>	Simulates a 'block interact' event. This is explained below.
/wg debug testdamage	[-t] <player>	Simulates an 'entity damage' event (use to test PvP and PvE). This is explained below.

Event Simulation

The event simulation commands are useful if, for example, players can't break blocks for some reason and you cannot identify the plugin causing it (without more drastic measures). Use of the `/wg debug testbreak` command would simulate the block break and record which, if any, plugins chose to block the virtual block break.

In order to use the commands, a player must be provided. The player is the *source* of the event. The target of the event (i.e. the block that being "broken" or the entity that is being "attacked"), however, is from the perspective of the person running the command. However, if `-t` is specified, then the target is from the perspective of the source player. The target is whichever block or entity is in the player's crosshair.

Several plugins may be listed in the output of the command, but only the first entry matters. This is because, for example, if Plugin A blocks the action, and Plugin B, which runs afterwards, "unblocks" the action, then Plugin A has no effect. WorldGuard lists the last running plugins first.

An example

If you want to see why PvP seems to be blocked, have another player (who is not able to PvP) look at you and then run the command:

```
/wg debug testdamage -t other_player_name
```

Be aware that the tests are not entirely complete. This is because Bukkit sometimes throws *other* events for some actions. For example, when filling a bucket, Bukkit fires a *bucket fill event*, which WorldGuard currently does not

provide a way to simulate. Lastly, sometimes Minecraft features (like adventure mode or the built-in spawn protection) may be at play.

> **Warning:** While the events are simulations in that Minecraft won't place or break the block in question, plugins do act upon the events. For example, a plugin may allow you to right click specially marked signs to teleport, and simulating an interact event on a teleport sign may possibly cause the victim player to be teleported.

1.4.5 Miscellaneous

Command	Parameters	Explanation
/wg version		Show WorldGuard's version.
/wg reload		Reload WorldGuard's configuration, blacklist, and region data.
/wg running		Show WorldGuard's running tasks. An example of a running task is a UUID lookup of a player that occurs in the background.

1.5 Blacklist

The blacklist allows an action (or several) to be performed whenever a player does something (place a block, use an item, etc.). Some example applications include:

- Blocking players from mining gold ore
- Notifying all administrators when a diamond block is found
- Telling the player a message when they place an enchantment table

An example of a blacklist file is:

```
# Deny lava buckets
[lavabucket]
ignore-groups=admins,mods
on-use=deny,tell
message=Sorry, you can't use lava buckets!

# Deny some ore
[goldore,ironore]
ignore-groups=admins
on-break=deny,tell,notify

# No TNT!
[tnt]
ignore-groups=admins
on-place=deny,notify,kick
```

1.5.1 Blacklist Files

For every world, you will find per-world configuration files:

- `worlds/world/blacklist.txt`
- `worlds/world_nether/blacklist.txt`

- `worlds/mining_world/blacklist.txt`

Hint: Unfortunately, you cannot share one blacklist file for several worlds from within WorldGuard. However, you may be able to create a "symlink" on your operaitng system to have two files refer to the same one.

1.5.2 The Format

A blacklist file consists of several sections in the format of:

```
[a list of items/blocks to match]
event to watch=what to do
event to watch=what to do
event to watch=what to do
option=value
```

Lines starting with # are ignored by WorldGuard. You can use this to place notes for yourself.

Matching Syntax

Items and blocks can be matched by either their (legacy) ID number or by specifying one of the entries on Bukkit's material list. Multiple items and blocks can be matched by separating each entry by a comma:

```
[wood,brick,glass]
```

1.5.3 Events

Now that you have specified the items or blocks to watch out for, you must specify the situations in which you would like to catch, as well as the action to perform.

The available events are listed below.

Event	Explanation
on-break	When the matched block is being mined
on-destroy-with	When the matched item or block is being held by the player during mining
on-place	When the matched block is being placed
on-use	When an item in the player's inventory is used
on-interact	When a player interacts (i.e. right click) with the matched block (door, lever, etc.)
on-drop	When the player drops the matched item
on-acquire	When the player acquires the matched item
on-dispense	When a dispenser dispenses the matched item

When specifying an event, a list of "actions" must be specified afterwards, as illustrated below:

```
[enchantment_table]
on-place=deny,tell
on-drop=notify
```

Available Actions

Multiple actions can be specified for each event.

Action	Explanation
deny	Denies the action (only if the blacklist **not** in "whitelist mode" (explained later))
allow	Permits the action (only if the blacklist is in "whitelist mode")
notify	Notify admins with the `worldguard.notify` permission
log	Log to console, file, or database
tell	Tell the player that the action was not allowed
kick	Kick the player
ban	Ban the player

1.5.4 Options

Options are specified in the same place as events, as illustrated below:

```
[enchantment_table]
on-place=deny,tell
message=Sorry, you can't use enchantment tables!
```

In this case, `message` is an option that overrides the message used by the "tell" action.

Option	Explanation
ignore-groups	Comma-separated list of permission groups to not affect
ignore-perms	Comma-separated list of permissions to not affect – make up your very own permissions!
comment	Message for yourself that is printed with `log` and `notify` actions (to override the default message)
message	Message to show the user (to override the default message). Put %s in the message to have it be replaced with the item name (in English)

1.5.5 Whitelist Mode

Switching the blacklist to whitelist mode (via the *Configuration*) will invert the behavior. Only entries with the `allow` action will be permitted.

Tip: Whitelist mode may seem very restrictive. To place a block, you would need to allow using the block (the item in the player's hand), interacting with the existing block (in the world, that's being clicked to place "on"), and placing the block. You may find *Build Permissions* easier to use since it supports wildcards (in permission nodes) depending on your use case.

1.5.6 Examples

Example: Blocking all bucket use

```
[lavabucket,waterbucket,bucket]
on-use=deny,tell
```

Example: Kicking players using TNT and notify administrators

```
[tnt]
ignore-groups=admins
on-place=deny,notify,kick
```

Example: Allowing only the people in the "obsidian" group and administrators to use obsidian

```
[obsidian]
ignore-groups=admins,obsidian
on-place=deny,tell
on-break=deny,tell
```

1.5.7 Logging

With the `log` action, messages can be logged to several places:

- Console
- File
- Database

These log targets can be enabled or disabled in the *Configuration*. Multiple log targets can be enabled at one time. By default, only the console log target is enabled.

Console Logging

Console logging merely prints the log entries to the server console.

File Logging

File logging writes the log entries to a file. In the *Configuration*, the path for the log file can be specified with special variables in it (like today's date), so you can have logs automatically rotated every day.

The following variables can be used:

- %Y the year (YYYY)
- %m the month (MM)
- %d the day (DD)
- %W the week of the year (00-52)
- %H 24-hour time (HH)
- %h 12-hour time (HH)
- %i the minute (mm)
- %s the second (ss)

- %u the user's name
- %% translates to a single percent sign "%"

Database Logging

WorldGuard can write the log entries to a MySQL database. However, you will have to create the database and table yourself first. The SQL needed to create the table is provided below:

```
CREATE TABLE IF NOT EXISTS `blacklist_events` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `world` varchar(10) NOT NULL,
 `event` varchar(25) NOT NULL,
 `player` varchar(16) NOT NULL,
 `x` int(11) NOT NULL,
 `y` int(11) NOT NULL,
 `z` int(11) NOT NULL,
 `item` int(11) NOT NULL,
 `time` int(11) NOT NULL,
 `comment` varchar(255) DEFAULT NULL,
 PRIMARY KEY (`id`)
);
```

1.6 Regions

WorldGuard lets you define named, 3-dimensional zones of various shapes called "regions." Each world can have zero or more regions defined, and each region may have owners, members, custom priorities, flags, and a parent region.

- Protect your spawn and other areas of your world
- Create a shopping district where players can own plots
- Prevent player versus player combat in neutral zones
- Deny entry into specific areas
- Setup hospital "heal" wards

A flexible system of parent-child hierarchy and priorities allows you to setup complex configurations.

1.6.1 Contents

Quick Start

Selecting Areas

To create a region, you need to tell WorldGuard the physical area. WorldEdit is used for this purpose.

New to WorldEdit?

You can read WorldEdit's quick start tutorial to learn how to select areas with WorldEdit.

Regions can take the shape of:

- Cuboids (i.e. a cube or a box)
- 2D polygons with heights

If you try to use an unsupported shape such as a cylinder, WorldGuard will inform you of this.

Basic Commands

Creating Regions

All regions must have a name, and names must be unique. Names are case-insensitive. Create a region with `/region define`:

```
/region define town
```

Rather than using `/region`, you can also use `/rg` as a shortcut:

```
/rg define town
```

The newly created region **automatically prevents building** within it. To let specific players build, add members or *owners* to the region. The main reason why you'd have separate owners or members is because you can set *permissions to use commands separately for owners and members*.

Either players or permission groups can be a member or owner. To specify a permission group, prefix the name of the group with `g:`. The relevant add and remove commands are illustrated below:

```
/rg addmember town Notch sk89q g:builders
/rg addowner town sk89q
/rg removemember town g:builders
/rg removeowner town sk89q
```

Full documentation for these commands can be found on the *Region Commands* page.

Lastly, when you create a region, you can also conveniently specify the owners:

```
/rg define town Notch sk89q g:builders
```

Region data is periodically saved, so you do not need to use a save command.

Example: Creating a spawn region where only the build team can build

1. Select the area of spawn.
2. Create a region named "spawn":

   ```
   /rg define spawn
   ```

3. Add the build team as a member:

   ```
   /rg addmember spawn g:builders
   ```

Hint: Your changes to regions will automatically be saved after a small delay. However, region data can be force saved using `/rg save`. All saves are done in the background and will not pause the server.

Other Commands

You can remove regions:

```
/rg remove town
```

You can list information about a region:

```
/rg info town
```

You can view a list of regions:

```
/rg list
```

You can change the shape of an existing region:

```
/rg redefine town
```

For more commands, see the *region commands* page.

Properties of Regions

Handling Overlap

Regions can overlap in WorldGuard, which provides a lot of flexibility but can be a source of complexity.

- When there are overlapping regions, a player must have permission to build in **all** overlapping regions.
- If you want a region to override another, use *priorities and region inheritance*.
- If you want a region to override all others in regards to whether players can build, use the `build` *region flag*.
- If a region you made isn't supposed to protect its area, use the `passthrough` *region flag*.

Example: Creating a free-for-all mining zone that overlaps spawn

1. Select the area of mining zone.
2. Create a region named "mine":

   ```
   /rg define mine
   ```

3. Set the `build` flag to `allow`, which permits building for all players and overrides spawn because spawn doesn't have the build flag explicitly set:

   ```
   /rg flag mine build allow
   ```

Flags

Each region can have extra settings applied to it called flags. For example, PvP can be blocked using the "pvp" flag:

```
/rg flag town pvp deny
```

Read the *Region Flags* guide for more information.

Example: Creating an arena at spawn where fighters can battle but cannot break blocks, while builders can still build within the arena

1. Select the area of arena.
2. Create a region named "arena":

```
/rg define arena
```

3. Set the `pvp` flag to `allow`:

```
/rg flag arena pvp allow
```

4. Builders cannot build in the arena because they aren't a member of the arena region, even if they are a member of the spawn region (if you have been following these examples). However, you created the arena region to set the PvP flag, not to protect the area, so set the `passthrough` flag to `allow` to make it transparent to build checks:

```
/rg flag arena passthrough allow
```

Common Scenarios

If you're wondering how to do something (like allow the usage of doors, levers, pressure plates, etc.), check out the *Common Scenarios* page.

Region Wand

The region wand lists all regions in a certain location. It a regular Minecraft item that can be right clicked on blocks. By default, the item is leather, but this can be changed in the *Configuration*.

In order to use the wand, the `worldguard.region.wand` permission must be assigned.

Region Flags

Regions can have flags set upon it. Some uses of flags include:

- Blocking player versus combat with the `pvp` flag
- Denying entry to a region using the `entry` flag
- Disabling the melting of snow using the `snow-melt` flag
- Blocking players within the region from receiving chat using the `receive-chat` flag
- Halting the growth of vines by using the `vine-growth` flag

A region can have several different flags set at one time, although a certain flag can only have one value at a time. Flags are defined using the `/region flag` command, as illustrated below for the "spawn" region and "hospital" regions:

```
/region flag spawn pvp deny
/region flag spawn greeting Welcome to spawn!
/region flag hospital heal-amount 2
/region flag hospital heal-delay 1
```

Remove a flag by not specifying a value:

```
/region flag spawn pvp
```

List flags by using the "flags" command:

```
/region flags spawn
```

The output of this command is interactive in-game. Click flag values to change them, and the arrows at the bottom to navigate through pages.

- *Region Groups*
- *Types of Flags*
- *Conflicting Flags*
- *Default Values of Flags*
- *Flag Listing*
  - *Overrides*
  - *Protection-Related*
  - *Mobs, Fire, and Explosions*
  - *Natural Events*
  - *Movement*
  - *Map Making*
  - *Miscellaneous*

Region Groups

Sometimes, it may be desired for a flag to only apply to a certain group of players rather than everyone that should enter the region. This can be achieved by specifying an additional "region group" when defining the flag, of which there are several options:

- all (everyone)
- members
- owners
- nonmembers
- nonowners

The group can be specified using the `-g` marker as illustrated below:

```
/region flag spawn -g nonmembers pvp deny
```

It is **not** possible to set the same flag to different values for more than one group on the same region. If you need that functionality, consider making several regions.

Note: When there are multiple overlapping regions, a player must be a member of the region *on which the flag is set* or *on one of the region's child regions* (when region inheritance is involved). This is explained further in *Priority and*

Inheritance.

Tip: The `entry` and `exit` flags default to "non-member", meaning setting them to "deny" will prevent non-members from entering/exiting the region. The `spawn` location flag defaults to "members", which means that only members can take advantage of it by default. The `nonplayer-protection-domains` flag has no region group. All other flags provided by WorldGuard default to "everyone".

Types of Flags

Each flag is of a certain type that determines what kind of values it may take. For example, the *heal-amount* flag is an numeric flag, so you can only set numeric values for it.

Type	Kind of values
state	Either 'allow' or 'deny' (explained later)
string	Any form of text
integer	A number that does not have decimals (5, but not 5.5)
double	Numbers that may have decimals (5, 5.5, 2.425)
location	A location in a world
boolean	True or false
set	A list of unique entries

Internally, there are more types, but it should generally not be of concern.

Tip: Most string flags will accept `\n` as a newline (for example, to send multiple lines via `greeting`/`farewell`, or a title and subtitle via `greeting-title` and `farewell-title`).

They may also accept color codes, either in the old style `&[0-9a-f]` or `` `[RrYyGgCcBbPp012w] `` for dark-red, red, dark yellow, yellow, etc., and `` [&`][klmnox] `` for obfuscated, bold, strikethrough, underline, and italic text.

They may also accept some replacements, such as `%name%` for the player's name, `%world%` for world name, and `%online%` for player count.

Example: Using string formatting options

Setting spawn's `greeting-title` to a fancy welcome message:

```
/rg flag spawn greeting-title `bWelcome to spawn!\n`YEnjoy your stay in `g`n%world
↪%`x, `C%name%`Y!
```

Conflicting Flags

Sometimes, a certain location may have multiple overlapping regions with different values for the same flag. The following rules are used to determine which values are selected:

- Regions will inherit the value of a flag from its parent, **if** the region did not have the flag set. (Note that the `build` flag is set implicitly with membership.)
- Higher priority regions will override lower-priority regions.

- The global region is considered like any other region, except it is at the lowest possible priority.

However, it is still possible for there to be conflicting flag values even after that process. Imagine two different regions at the same priority, for example. At that point, the value of the flag is decided differently depending on the type of flag:

- For state flags, if `deny` is present, the result is `deny`. Otherwise, if `allow` is present, then the final value is `allow`.
- For other flags, the result is not defined. For that reason, do not, for example, set two different greeting messages in the same area with the same priority.

Default Values of Flags

Flags can have default values, which are used if a flag is not set for the subject's region group on any region that affects the action. The default behavior is whichever is most sensible. For example, if `item-pickup` is not defined, WorldGuard defaults to allowing it. You can view the default values ingame by using the "flags" command as already described above. State flags can either be allowed by default, or have no default value. A rule of thumb is that protection-related flags, as listed below, have no default value, except for the `build` flag. All other state flags are usually allowed by default. The `build` flag is special. Its default value depends on the region membership of the subject and it's always set implicitely, if it's not set explicitely on any highest priority region that affects an action. That's the reason why members and owners can build and non-members can't by default. All other protection-related flags are always checked in tandem with the build flag. Thus their default behavior is the resulting value of the build flag.

Tip: Most of the time you don't need to set any flags. You should only set flags if you want to deviate from the default values or to overwrite set flags of other regions. Specifically don't set protection-related flags to completely protect or unprotect a region. Use membership or the `passthrough` flag instead.

Flag Listing

Flags are broken down into categories below.

Overrides

Flag	Type	description
passthrough	state	This flag is short for 'passthrough build'. It has nothing to do with movement. • If not set **(default)**, then the region protects it area. • If set to `deny`, then the region protects its area. • If set to `allow`, then the region **no longer** protects its area. Where does the flag come into use? • When you are using other flags (PvP, healing, etc.) and you don't want to prevent building. • Why not set `build` to `allow` (explained later) instead? That would override other regions and let people build!
nonplayer-protection-domains	set of strings	Non-player associables, such as pistons, are usually members of either all regions in which they are in or only of the regions with the highest priorities in which they are in, depending on the `use-max-priority-association` *Configuration* setting and if it is a *Global Region*. Thus there can be borders between regions, such that pistons cannot move blocks from one region into another region. The borders between the regions can be removed for non-player associables by setting the `nonplayer-protection-domains` flags of these regions. This flag is a set of strings, describing the domains to which the region belongs. If a non-player associable is a member of a region, then it is also a member of another region, if there is at least one domain to which both regions belong to.

Protection-Related

Flag	Type	description
build	state	Everything: • Whether blocks can be mined or placed • Whether doors, levers, etc. (but not inventories) can be used • Whether entities and blocks can be interacted with • Whether player versus player combat is permitted • Whether sleeping in a bed is permitted • Whether inventories can be accessed • Whether vehicles (boats, minecarts) can be placed • etc.
interact	state	Everything that involves 'using' a block or entity: • Whether doors, levers, etc. (but not inventories) can be used • Whether vehicles (including animals) can be mounted • etc.
block-break	state	Whether blocks can be mined
block-place	state	Whether blocks can be placed
use	state	Whether doors, levers, etc. (but not inventories) can be used
damage-animals	state	Whether players can harm friendly animals (cows, sheep, etc)
chest-access	state	Whether inventories can be accessed
ride	state	Whether vehicles (including animals) can be mounted
pvp	state	Whether player versus player combat is permitted
sleep	state	Whether sleeping in a bed is permitted
respawn-anchors	state	Whether respawn anchors can be activated
tnt	state	Whether TNT detonation or damage is permitted
vehicle-place	state	Whether vehicles (boats, minecarts) can be placed
vehicle-destroy	state	Whether vehicles can be destroyed
lighter	state	Whether flint and steel can be used
block-trampling	state	Whether farmland and turtle eggs can be trampled
frosted-ice-form	state	Whether players with frost walker boots will form ice
item-frame-rotation	state	Whether items can be rotated within item frames
firework-damage	state	Whether fireworks can deal damage to entities
use-anvil	state	Whether anvils can be used
use-dripleaf	state	Whether dripleaf can be used

Warning: None of these flags are player-specific. For example, the `block-break` flag, if set to deny, **prevents pistons from breaking blocks**.

To understand why, consider the fact that players can fling TNT into a region from outside, or a player can build an inchworm piston machine that moves into another region. While these actions were caused by a player, realistically attempting to figure which player built the TNT cannon or used it is much more difficult. However, you still want to prevent someone from blowing up spawn with a TNT cannon.

Outright blocking TNT cannons or pistons is the wrong solution. Pistons and TNT cannons should be allowed in *some* cases. For example, a TNT cannon or piston inside should work *within* the region.

First off, remember who can build in regions: it's **not** players, it's **members**. When we consider pistons or TNT, it should be no different. How does WorldGuard figure out whether a piston machine or TNT cannon is a member of a region? **If it's inside the region,** of course!

When you create a region, before setting any flags on it:

- Members may build
- Non-members may **not** build

TNT cannons and pistons inside are allowed to work because they are "members." An imaginary player, "Bobby," who isn't a member yet, is unable to place or break blocks. Once you add Bobby to the region, then Bobby can build.

When you set the protection flags, you override this behavior. If you set `block-break` to `deny`, then even members are unable to break blocks. Bobby cannot break blocks. A TNT cannon inside cannot break blocks. A piston inside cannot break blocks. **You break pistons.**

That raises two questions:

- **How do I prevent players from placing or breaking blocks?** Don't do anything. Don't change any flags! Remember, only members can build by default.
- **How do I change a flag to only affect players?** You probably mean: how do you make a flag only affect *non-members*? Well, that's easy: use *Region Groups*.

Tip: Note: If the `build` flag is set to `allow` or `deny`, it can still be overriden with a different flag (`block-break`, `interact`, etc.). The `build` flag is set implicitly with membership.

Mobs, Fire, and Explosions

Flag	Type	description
creeper-explosion	state	Whether creepers can do damage
enderdragon-block-damage	state	Whether enderdragons can do block damage
ghast-fireball	state	Whether ghast fireballs and wither skulls can do damage
other-explosion	state	Whether explosions can do damage
fire-spread	state	Whether fire can spread
enderman-grief	state	Whether endermen will grief
snowman-trails	state	Whether snowmen will create snow beneath them
ravager-grief	state	Whether ravagers will grief
mob-damage	state	Whether mobs can hurt players
mob-spawning	state	Whether mobs can spawn
deny-spawn	set of entity types	A list of entity types that cannot spawn
entity-painting-destroy	state	Whether non-player entities can destroy paintings
entity-item-frame-destroy	state	Whether non-player entities can destroy item frames
wither-damage	state	Whether withers can do damage (with their body explosions - skull projectiles are handled by ghast-fireball as mentioned above)

Example: Preventing sheep and cows from spawning at spawn

The entity types must be specified:

```
/rg flag spawn deny-spawn cow,pig
```

Natural Events

Flag	Type	description
lava-fire	state	Whether lava can start fires
lightning	state	Whether lightning can strike
water-flow	state	Whether water can flow
lava-flow	state	Whether lava can flow
snow-fall	state	Whether snow will form tiles on the ground
snow-melt	state	Whether snow will melt
ice-form	state	Whether ice will form
ice-melt	state	Whether ice will melt
frosted-ice-melt	state	Whether frosted ice will melt
mushroom-growth	state	Whether mushrooms will grow
leaf-decay	state	Whether leaves will decay
grass-growth	state	Whether grass will grow
mycelium-spread	state	Whether mycelium will spread
vine-growth	state	Whether vines (and kelp) will grow
rock-growth	state	Whether rocks (dripstone, etc) will grow
sculk-growth	state	Whether sculk (sculk and sculk vines) will grow
crop-growth	state	Whether crops (wheat, potatoes, melons, etc) will grow
soil-dry	state	Whether soil will dry
coral-fade	state	Whether coral will die when not in water.

Warning: The `fire-spread`, `water-flow`, `lava-flow`, and `leaf-decay` flags require that the "high frequency flags" option be enabled in the *configuration*. This is because these events can be very frequent, requiring more region lookups, and potentially slowing down your server (or at least warming the server room a bit more).

Movement

Flag	Type	description
entry	state	Whether players can enter the region
exit	state	Whether players can exit the region
exit-via-teleport	state	Whether players can exit the region via teleport. This only takes effect if the player is otherwise denied exiting the region
exit-override	boolean	Whether to always allow a player to exit
entry-deny-message	string	The message issued to players that are denied entry
exit-deny-message	string	The message issued to players that are denied exit
notify-enter	boolean	Whether players with the `worldguard.notify` permission are notified when another player enters the region
notify-leave	boolean	Whether players with the `worldguard.notify` permission are notified when another player leaves the region
greeting	string	The message that appears in chat upon entering the region
greeting-title	string	The title that appears upon entering the region. Including a newline (`\n`) will send a subtitle.
farewell	string	The message that appears in chat upon leaving the region
farewell-title	string	The title that appears upon leaving the region. Including a newline (`\n`) will send a subtitle.
enderpearl	state	Whether enderpearls can be used
chorus-fruit-teleport	state	Whether chorus fruits can be used to teleport
teleport	location	The location to teleport to when the `/region teleport` command is used with the region name
spawn	location	The location to teleport to when a player dies within the region
teleport-message	string	The message issued to players that are teleported with `/region teleport`

Tip: As mentioned above, the `spawn` location flag defaults to "members", which means that only members can take advantage of it by default. Set the region group for the flag to change this.

Tip: If overlapping regions have the same `greeting` or `farewell` flag, no message is sent when moving between these regions, e.g. if you enter one region while being in the other. This is also true, in a more general sense, of any player movement that does *not* result in the flag at the "from" and "to" locations changing.

Warning: The `greeting` and `farewell` message flags require that the "use player move event" option **not** be disabled in the *configuration*.

Example: Preventing non-members of a "secret_club" region from entering it

The key is to set the region group to "nonmembers":

```
/rg flag secret_club entry -g nonmembers deny
```

Map Making

Flag	Type	description
item-pickup	state	Whether items can be picked up
item-drop	state	Whether items can be dropped
exp-drops	state	Whether XP drops are permitted
deny-message	string	The message issued to players that are denied an action
invincible	state	Whether players are invincible
fall-damage	state	Whether entities receive fall damage
game-mode	gamemode	The gamemode (survival, creative, adventure) that will be applied to players that enter the region
time-lock	string	Time of day in ticks (between 0 and 24000) that players will see the world as while in the region. Use + or - for time relative to the world time.
weather-lock	weather	Type of weather players will see when in the region. This does not affect world mechanics. Valid values are `rain` and `clear`.
natural-health-regen	state	Whether players should naturally regen health from being satiated or being in peaceful mode.
natural-hunger-drain	state	Whether players should naturally lose hunger due to saturation/exhaustion levels.
heal-delay	integer	The number of seconds between heals (if `heal-amount` is set). Set to 0 to disable.
heal-amount	integer	The amount of half hearts to heal (or hurt if negative) the player at the rate of `heal-delay`
heal-min-health	double	The minimum number of half hearts that damage (via `heal-amount`) will not exceed
heal-max-health	double	The maximum number of half hearts that healing (via `heal-amount`) will not exceed
feed-delay	integer	See equivalent heal flag, except this is for food
feed-amount	integer	See equivalent heal flag, except this is for food
feed-min-hunger	integer	See equivalent heal flag, except this is for food
feed-max-hunger	integer	See equivalent heal flag, except this is for food
blocked-cmds	set of strings	A list of commands to block
allowed-cmds	set of strings	A list of commands to whitelist (any unallowed commands will be blocked)

Warning: The healing and feeding flags require that the "use player move event" option **not** be disabled in the *configuration*.

Example: Changing the message players receive when an action they try is blocked

Set the `deny-message` flag:

```
/rg flag spawn deny-message Sorry! You are at spawn. If you want to find a place to␣
↪call home, use the rail station to leave spawn.
```

Example: Blocking the "/tp" and "/teleport" commands at spawn

The commands in question can be blocked with:

```
/rg flag spawn blocked-cmds /tp,/teleport
```

Example: In a "hospital" region, heal players one heart every second up to half their health bar

Without any buffs, the player's maximum health is 20, so 10 is half of that:

```
/rg flag hospital heal-delay 1
/rg flag hospital heal-amount 2
/rg flag hospital heal-max-health 10
```

Miscellaneous

Flag	Type	description
pistons	state	Whether pistons can be used
send-chat	state	Whether players can send chat
receive-chat	state	Whether players can receive chat
potion-splash	state	Whether potions can have splash effects

Priority and Inheritance

As noted in the *Quick Start*, regions can overlap. When there are overlapping regions, a player must have permission to build in **all** overlapping regions. With priorities and inheritance, however, that rule does not have to be satisfied.

Priorities

Every region defaults to a priority of 0, but it can be adjusted to be positive or negative. Higher numbers imply higher priorities. The valid range of priorities is -2147483648 to 2147483647, inclusive, but you will probably use more reasonable numbers like -2, 10, 15, or 100.

- In terms of membership and who can build: When a certain location has overlapping regions, only the regions of the highest priority are considered.
- In terms of *region flags*, the highest priority region with the flag defined is used.

Priorities can be set with `/rg setpriority`:

```
/rg setpriority example 5
```

Example: Creating an "pub" region at spawn where only those in the "pub group" can build, even though spawn has already has region where "builders" can build

If you simply make a region overlapping spawn, then a player would have to be both a member of "pub" *and* "builders" in order to build within the pub. Therefore, you want to make the pub region higher priority.

1. Select the pub area.

2. Create the "pub" region:

```
/rg define pub
```

3. Raise the priority of "pub" to some number higher than 0, the priority of the spawn region:

```
/rg setpriority pub 10
```

Example: Making a special "heal" area where PvP is denied within an "arena" region where PvP is permitted

The goal here is to override the `pvp = allow` flag of the arena.

1. Select the heal area.
2. Create the "heal" region:

```
/rg define heal
```

4. Set the `pvp` flag to `deny`:

```
/rg flag heal pvp deny
```

3. Raise the priority of "heal" to some number higher than 0, the priority of the arena region:

```
/rg setpriority heal 10
```

Note: Since *deny* overrides *allow* when there is a conflict, you don't actually need to raise the priority of "heal." However, you would need to set the priority if you wanted to override *deny* with *allow*.

Inheritance

Following from the pub example above, what if you want to allow *both* builders and pub members to build in the pub? You could add the builders group to the pub region, or instead, you could have the pub *inherit* its members from the spawn region.

When a region is given a parent:

- It inherits the parent's members and owners
- It inherits the parent's flags *if* if the flag isn't defined on the child region

This comes in handy for:

- A main area and small plots inside this main area
- Creating a template region by which child regions will inherit the flags

Every region can have at most one parent. Set parents with `/rg setparent`:

```
/rg setparent region_name parent_name
```

To remove a parent, don't specify a parent name:

```
/rg setparent region_name
```

WorldGuard will detect circular inheritance and prevent it.

Example: Creating a "mall" with multiple plots inside of it

Once the mall, plot1, and plot2 regions have been created, parents can be set on the plots:

```
/rg setparent plot1 mall
/rg setparent plot2 mall
```

Let's say you have a "mall_owners" group that you add to the mall region:

```
/rg addowner mall g:mall_owners
```

The mall owners will be able to build within the plots.

If you add a member or owner to one of the plots, the player will be able to build only within their plot:

```
/rg addowner plot1 sk89q
```

Note: Non-player associables, such as pistons, can also be members of regions. Member inheritance is not limited to players only. This means that pistons in parent regions, for example, can push blocks into their children.

Template Regions

As previously mentioned, because flags are inherited from their parents, a parent region can act as a base template for all of its child regions.

However, you may want the template region to *not actually physically exist*, since you're not using it to protect an area. One way to do this is to create a "global region," which is a region that has no actual physical size. Create global regions with the `-g` switch on the region creation command:

```
/rg define -g plot_template
```

Example: Having the plots extend a "plot_template" region in the mall example

Create the plot template region:

```
/rg define -g plot_template
```

Have the plots inherit from the plot template:

```
/rg setparent plot1 plot_template
/rg setparent plot2 plot_template
```

Have the plot template inherit from mall:

```
/rg setparent plot_template mall
```

Let's say you want to let players access any chest in the mall, *except* those in plots. First, you'd use the `chest-access` flag on the mall region:

```
/rg flag mall chest-access allow
```

However, now you need to deny `chest-access` in every plot. Fortunately, you have the plot template that you can use:

```
/rg flag plot_template chest-access deny
```

Inheritance versus Priorities

A higher priority parent region will override its children, so inheritance only properly works when the children are of the same or higher priority compared to their parent regions.

Region Groups and Overlapping Regions

As you may be aware, flags can be made to apply to only certain groups:

```
/rg flag mall pvp -g nonmembers deny
```

When there is only one region, it's clear which players are members and which are not. However, it's less obvious when there are overlapping regions: does a player only need to be a member of one of the regions?

The answer is no. The player must be a member of the region *on which the flag is set*.

For example, let's imagine two overlapping regions:

- Spawn, with flag `pvp -g nonmembers deny` and no members
- Market, with member "sk89q"

PvP would be denied for sk89q because sk89q is not a member of spawn.

In the Context of Inheritance

When a region has a parent set, both flags and the list of members are inherited, so a player *can* be a member of one of the child regions too.

For example, if there are two regions:

- Market, with flag `pvp -g nonmembers deny` and no members
- Shop1, with member "sk89q," **inheriting** from the market

Is sk89q a member? Yes, so sk89q is allowed to PvP.

Note: As mentioned previously, parent regions should **not** have a higher priority than their children, otherwise this will not work correctly. PvP would be denied for sk89q because the parent region would override the child.

Global Region

The global region is a special region that:

- Encompasses the entire world
- Has the lowest possible priority
- Has some special behavior

Every world has its own global region. However, the global region does not exist until you attempt to modify it by referring to the global region with the name `__global__`.

For example, setting a flag on the global region would create the region automatically:

```
/rg flag __global__ pvp deny
```

Properties

Think of the global region as a region that is always there. Flags set on the global region, aside from a few flags, work like they would on any other region.

However, unlike normal regions, the `passthrough` *region flag* is implicitly set to *allow*. If you recall, setting passthrough to allow makes the region a non-protection region, so players can build in the region as long as there are not other regions preventing it.

Changing `passthrough`

If you change the `passthrough` *flag* to deny, making the global region act like a regular region, players must be owners or members of the global region in order to build in it. Because the global region encompasses the entire world, that means building is denied by default.

Because the global region is considered the lowest priority region, any other normal regions made will override the global region and thus allow building if a player is a member of that region.

Note: The passthrough flag has nothing to do with movement. It means "passthrough of build permissions," as better explained in *Region Flags*.

Example: Preventing building in the "wilderness"

The `passthrough` flag can be set to `deny`:

```
/rg flag __global__ passthrough deny
```

Adding Members

Due to legacy reasons, adding owners or members to the global region implicitly sets `passthrough` to deny. That means that you do not have to actually change the flag, although it is recommended anyway.

Non-Player Associables

Non-player associables, such as pistons, are usually members of either all regions in which they are in or only of the regions with the highest priorities in which they are in, depending on the `use-max-priority-association` *Configuration* setting. However, non-player associables are no longer members of the global region if they are in at least one other region. This means that pistons, for example, cannot push blocks from inside a region into a protected global region. Thus the global region always behaves as if `use-max-priority-association` is set to `true`.

The membership of non-player associables can additionally be adjusted by setting the `nonplayer-protection-domains` flags of the regions. However, this flag works differently on the global region. If there is at least one domain to which a normal region and the global region belong to, this implies that non-player associables that are members of the normal region, will also be members of the global region but **not** vice versa.

Build Flag

Setting the `build` flag to `allow` has **no** effect as there should be no reason to. Setting the flag to `deny` works like it does with any other region, but be aware that setting `build` to deny on any region essentially means that *nothing* can break or place blocks, therefore breaking things like pistons. Since the global region encompasses the entire world, it would break all pistons that are not within another region.

> **Warning:** Setting the `build` flag on the global region is strongly discouraged. If you want to protect the world by default, set the `passthrough` flag.

Overriding Defaults

Flags in WorldGuard come with certain *defaults*. For example, the `exp-drop` flag defaults to allow if no regions set it, which means that experience drops are permitted even when a player is not a member or owner of the region.

If you want to override the `exp-drops` flag by setting it to `deny` and having it apply to only non-members, setting it on the global region would not work. For example, you may be tempted to use `/rg flag __global__ exp-drop -g nonmembers deny`, but this will **not** work. When you specify non-members, it refers to non-members of the *global region*. Thus, if you make, for example, a plot region, XP drops would be denied for the plot's members because the plot's members are not members of the global region. (Remember, the global region is of a low priority. However, regions do not inherit from it so flags do not propagate.)

In these cases, it is recommend that you create a "template region" as explained *in the priorities and inheritance page*.

Region Commands

For a list of other (non-region protection related) commands, see the *Commands* page. Permissions for these commands is detailed on the *Permissions* page.

Some of these commands may run in the background and then later return results. The list of active background commands can be viewed with `/wg running`. There is a hard limit to the number of running and queued commands.

Either `/region` or `/rg` can be used for commands. Some sub-commands have multiple aliases (for example, `/rg d` can be used a shortcut to `/rg define`).

Tip: You can use almost all of these commands from console, but you may need to specify the `-w` flag (when available).

You can also use the //world command from WorldEdit to set the target world (both from console and in-game).

Creating and Removing Regions

Define

```
/rg define [-w <world>] [-g] <id> <owner1> [<owner2>] [... <ownerN>]
/rg create (...)
/rg d (...)
```

Creates a new region with a given ID and an optional list of owners. Your current WorldEdit selection is used for the area of the region.

Region IDs are case-insensitive. Only one region can exist with a given name (per-world).

- `-g` will create a new "global" region (not the same as the *Global Region*) that has no physical space, which is useful for creating template regions (see *Priority and Inheritance*)

Example: Creating a new "shop" region with "sk89q" and "wizjany" as owners

```
/rg define shop sk89q wizjany
```

Remove

```
/rg remove [-w <world>] [-f] [-u] <id>
/rg rem (...)
/rg delete (...)
/rg del (...)
```

Removes a region.

If the specified region has child regions, then either `-u` or `-f` must be specified. Both options cannot be specified together.

- `-u` changes child regions of the specified region to have no parent
- `-f` removes child regions of the specified region

Redefine

```
/rg redefine [-w <world>] [-g]
/rg update (...)
/rg move (...)
```

Changes the physical area associated with an existing region and replaces it with your current WorldEdit selection.

- `-g` will create a new "global" region (not the same as the *Global Region*) that has no physical space, which is useful for creating template regions (see *Priority and Inheritance*)

Claim

```
/rg claim <id>
```

Claims a region, which is for self-serve player-created regions. See *Claiming* for more information.

Editing Memberships

Add Member

```
/rg addmember [-w <world>] <id> <members...>
/rg addmem (...)
/rg am (...)
```

Adds any number of members to a region. Using `g:<member>` will add a permission group instead of a player.

Example: Adding the "builder" group and the player "sk89q" as members of a "spawn" region of the "lobby" world.

```
/rg addmember -w lobby spawn g:builder sk89q
```

Add Owner

```
/rg addowner [-w <world>] <id> <owners...>
/rg ao (...)
```

Adds any number of owners to a region. Using `g:<owner>` will add a permission group instead of a player.

Example: Adding the "admins" group and the player "eduardo" as members of a "spawn" region of the "lobby" world.

```
/rg addowner -w lobby spawn g:admins eduardo
```

Remove Member

```
/rg removemember [-w <world>] [-a] <id> <members...>
/rg remmember (...)
/rg remmem (...)
/rg rm (...)
```

Removes any number of members from a region. As in the add command, use `g:<member>` to specify a permission group.

- `-a` will remove all members from the region, ignoring the <members...> argument

Remove Owner

```
/rg removeowner [-w <world>] [-a] <id> <owners...>
/rg ro (...)
```

Removes any number of owners from a region. As in the add command, use `g:<owner>` to specify a permission group.

- `-a` will remove all owners from the region, ignoring the <owners...> argument

Getting Information

Select

```
/rg select <id>
/rg sel (...)
/rg s (...)
```

Replaces your current WorldEdit selection with the area of an existing region.

Information

```
/rg info [-u] [-s] [-w <world>] [<id>]
/rg i (...)
```

Displays information about a specified region, or if no region is specified, the region that you are currently in. If you are in several regions, then a list will shown instead.

- `-s` causes the command to select the region (see `/rg select`)
- `-u` causes UUIDs to be shown rather than player's last seen names

Example: Showing information about the Global Region

```
/rg info __global__
```

Flags

```
/rg flags [-w <world>] [-p <page>] <id>
```

Displays a paginated, interactive list of flags for the given region.

Explicitly set flags are shown with white values, flags inherited from a parent region are shown in light gray, and unset flags are shown with their default value in dark gray.

Clicking on the values allows you to quickly set and unset flags, and the arrows at the bottom next to the page number can be used to navigate the list.

List

```
/rg list [-i <id search>] [-p <player>] [-w <world>] [<page>]
```

Lists the regions that have been created. A number can be provided to show a certain page.

If a player doesn't have permission to list all regions but has permission to list their own (ones the player is a member or owner of), then the command will automatically only list the player's own regions.

- `-p <player>` can be specified to filter on regions that the given player is a member or owner of
- `-i <id search>` can be specified to filter region IDs containing the search text
- `-s` can be specified to match only regions which physically intersect your WorldEdit selection

Example: Listing regions that "sk89q" is a member or owner of

```
/rg list -p sk89q
```

Setting Region Options

Flag

```
/rg flag <id> <flag> [-w <world>] [-g <group>] [-e] [<value>]
```

Sets a flag on a region (see *Region Flags* for more information).

To unset a flag, don't specify a value.

To set a flag to a blank value, use `-e`. This is useful for setting flags like `greeting` to a blank message to override the flag set in a different, larger, and lower priority region. If `-e` is specified in addition to a value, the value is discarded.

- `-g <group>` specifies the region group (see *Region Flags*)
- `-e` sets an empty value

Example: Setting the `pvp` flag of "mall" to "deny" with region group "nonmembers"

```
/rg flag mall pvp -g nonmembers deny
```

Example: Unsetting the `greeting` flag on "mall"

```
/rg flag mall greeting
```

Example: Setting the `greeting` flag to an empty value

```
/rg flag mall greeting -e
```

Priority

```
/rg setpriority [-w <world>] <id> <priority>
/rg priority (...)
/rg pri (...)
```

Sets the priority of a region. See *Priority and Inheritance* for more information.

The default priority of a region is 0.

Parent

```
/rg setparent [-w <world>] <id> [<parent>]
/rg parent (...)
/rg par (...)
```

Sets the parent of a region. See *Priority and Inheritance* for more information.

To unset a parent priority, specify no parent.

Example: Setting the parent of "plot1" to "mall"

```
/rg setparent plot1 mall
```

Example: Removing the parent of "plot1"

```
/rg setparent plot1
```

Miscellaneous Commands

Teleport

```
/rg teleport [-c] [-s] <id>
```

Teleports yourself to the location specified by either the `spawn` or `teleport` *flags*.

- `-s` selects the spawn flag rather than the teleport flag
- `-c` teleports you to the geometric center of the region even if neither flag is set. This requires you to be in spectator mode.

Management Commands

Load

```
/rg load [-w <world>]
/rg reload (...)
```

Reloads the region data from file or database. If recent changes were made in-game to the region data, this may cause data loss.

The load operation occurs in the background and will not pause the server. If the command is used before a previous load has completed, the new load will be queued. There is a limit to the maximum number of operations that can be queued.

Save

```
/rg save [-w <world>]
/rg write (...)
```

Saves the region data to disk.

Tip: Region data is saved automatically soon after any changes are made, so this command does not need to be called explicitly.

The save operation occurs in the background and will not pause the server. If the command is used before a previous save has completed, the new save will be queued. There is a limit to the maximum number of operations that can be queued.

Migrate Database

```
/rg migratedb <from> <to>
```

Migrates from one type of storage driver (see *Storage Drivers*) to another.

Valid choices for "from" and "to" are:

- `yaml`
- `mysql`

Migration does not automatically enable the target storage driver – that must be done in the *Configuration*.

Warning: Be sure to make a backup before running migration.

Warning: This command does not run in the background and will pause the entire server. If your server software has server pause detection, this may kill the server during migration and abort the migration process. If migration is aborted or fails, you may need to empty the target storage before re-running migration.

Migrate UUID

```
/rg migrateuuid
```

Converts player names in the region data to Mojang UUIDs.

Names that have no corresponding UUIDs will either be removed or left remaining depending on the *Configuration* (the setting is `keep-names-that-lack-uuids`).

Warning: Be sure to make a backup before running migration.

Warning: This command does not run in the background and will pause the entire server. If your server software has server pause detection, this may kill the server during migration and abort the migration process.

Migrate Region Heights

```
/rg migrateheights
```

Extends regions that were physically defined from min y <= 0 to max y >= 255 (i.e. the world limits pre-MC 1.18) to the new world height limits.

Useful for updating from 1.17 or before to 1.18 or later if you had a lot of regions which were defined at the world limits. Note that if you intentionally had regions beyond the world limits before 1.18, you will need to move them beyond the new world limits manually.

Warning: Be sure to make a backup before running migration.

Warning: This command does not run in the background and will pause the entire server. If your server software has server pause detection, this may kill the server during migration and abort the migration process.

Claiming

A rudimentary player-oriented claiming system is available for use in WorldGuard. It creates regions like `/rg define` would, but players must use a special command with separate permissions. WorldEdit selections are still required for selecting the physical area of the region, although only the `worldedit.selection` permission is necessary for that.

Player can claim selected regions using the `/rg claim` command:

```
/rg claim region_name
```

The player running the command will automatically be added as a region owner. Access to the command is provided by the `worldguard.region.claim` permission.

Rules

- **Maximum region count:** Unless the player has the `worldguard.region.unlimited` permission, the player cannot exceed the maximum number of regions that is permitted in the *Configuration* for the player.
- **Maximum region volume:** Unless the player has the `worldguard.region.unlimited` permission, the volume of a claim must not exceed the `regions.max-claim-volume` configuration.
- **Overwrite prevention:** The region to be claimed cannot replace an existing one with the same name.
- **Overlap prevention:** The region to be claimed cannot overlap another region that the player is not an owner of.
- **Overlap only owned:** If `regions.claim-only-inside-existing-regions` is enabled, the region must overlap another region that the player is an owner of. Note that the region to be claimed does *not* need to be fully contained within another region (see WORLDGUARD-2689).

Note: At this time, polygonal regions are not supported. See WORLDGUARD-3173.

Other Commands

You can give players the ability to use other commands (like setting flags and adding members) by setting the appropriate *Permissions*.

Storage Drivers

Region data can be stored using either:

Storage Driver	Advantages
YAML	• No database server required. • Can simply modify the file to edit the region data manually. • Easy to backup with the world. • Fast full load and full save.
MySQL	• Can access data from other systems more easily. • When saving, saves only changed data.

Note: Other SQL-supporting databases (PostgreSQL, SQL Server, SQLite, etc.) are currently not supported.

Warning: SQL support for region storage was contributed by a third-party developer and despite large rewrites, there are major issues with using it. It is highly recommended to never use SQL for storing your WorldGuard regions.

If you are already using SQL for region storage, you should migrate back to YAML using the migrate command covered below. SQL support for region storage will be removed from WorldGuard in a future version.

Switching Drivers

The default mode is to use YAML files.

In the *Configuration*, `regions.sql.use` can be enabled to switch to MySQL, therefore disabling YAML. If you are switching to MySQL, the database tables will be created automatically, though be sure to provide the necessary permissions for the SQL user account.

Warning: It is highly recommended that you make a backup before switching drivers.

Migrating Data

If you simply switch between two different storage drivers, you will end up with no regions because the new storage medium will have no data. However, you can use a special migration command to migrate the data between two storage mediums, and you are able to do this before **or** after switching to the new driver.

To migrate data, use: `/rg migratedb from to`, specifying the source and target driver. For example, to migrate from YAML to MySQL, you would use:

```
/rg migratedb yaml mysql
```

Please make sure that the target database is empty.

Storage Drivers

Warning: Remember to make backups if you manually modify data outside WorldGuard.

YAML

Region data stored in YAML is found within WorldGuard's configuration folder as "region.yml" in each world folder. The file can be modified while the server is running and reloaded using `/rg load` without issue.

The region file is structured like the following example:

```
regions:
    test:
        min: {x: 1730.0, y: 0.0, z: -169.0}
        max: {x: 1742.0, y: 255.0, z: -158.0}
        members:
            players: [bobby]
            unique-ids: [0ea8eca3-dbf6-47cc-9d1a-c64551ca975c]
        flags: {use: allow, greeting: Welcome!, pvp: allow, pvp-group: MEMBERS}
        owners:
            groups: [admins]
        type: cuboid
        priority: 4
    __global__:
        members: {}
        flags: {}
        owners: {}
        type: global
        priority: 0
```

UUIDs are normally stored, but player names can also be added using `-n` on some of the commands.

MySQL

- Only one server instance can use one set of tables.
- A table prefix setting is provided to use the same database for different data sets or installations.
- Changes are made in a transaction and will be rolled back if an error occurs.
- By default, WorldGuard will only save changes to region data rather than re-save the entire region data set.

The tables used are explained below:

Table	Purpose
region	Region data, with shape, priority, and parent information.
region_cuboid	Data for cuboid regions, with bounds.
region_poly2d	Data for polygonal regions, with minimumum and maximum Y values.
region_poly2d_point	Individual rows for points of polygonal regions.
region_flag	Per-region flag data.
region_players	List of players on regions.
region_groups	List of groups on regions.
world	Normalizes worlds into a world ID.
user	Normalizes users into a user ID.
group	Normalizes groups into a group ID.

Each user row will either have a UUID or name set.

Warning: Modification of the data stored in MySQL while WorldGuard is running on a server is not recommended. Because WorldGuard will save only changes to region data, it may cause problems if external modifications change the state of the region data in a way that WorldGuard is unable to expect.

High-Latency Environment

Data is loaded and saved in its entirety and in the background, so a slow hard disk or a remote MySQL server should not pose too much issue.

What's Protected?

Protection is extremely comprehensive:

- Block break and placement are prevented.
- Inventories (chests, furnaces) cannot be opened.
- TNT cannons are blocked (from outside a region).
- Gravel cannons are blocked (from outside a region).
- Tree growth grief (from outside a region) is prevented.
- Pistons are blocked (blocks cannot be pulled from a protected region nor pushed into).
- Sign change exploits are prevented.
- Crop trampling is prevented.
- Minecarts and boats are protected.
- Paintings and item frames are protected.
- Doors, buttons, pressure plates, and levers are protected.
- Animals are protected (leashing, taming, shearing).
- Unauthorized splash potions and projectiles are prevented.

In addition, WorldGuard supports mod-added custom blocks and entities (with some exceptions, explained later).

Water flow and lava flow protection is disabled by default, but it can be enabled in the *Configuration*.

Tip: If you find a way to bypass protection, please file a ticket on the issue tracker.

Exceptions

Some sensible defaults are used that permit certain activities from non-members:

- Item drops and item pickups are permitted.
- XP drops are permitted.

These exceptions can be removed globally or per-region by adjusting *Region Flags*.

However, hoppers are one exception that cannot be changed. A hopper adjacent to a protected region can place items into a chest inside the region, so chests should not be placed on the edge of regions. This behavior is used because the preivous situation (where hoppers were protected) was a source of frequent confusion, though the exception may be removed in the future.

Additional exceptions can also be added per-region by adjusting flags, or globally by using the `interaction-whitelist` option in the *Configuration*.

Example: Allowing everyone to use doors, levels, buttons, and pressure plates even in protected regions

The `use` flag concerns these types of items:

```
/rg flag __global__ use allow
```

See the *Global Region* page for more information about `__global__`.

Example: Blocking item drops and item pickups at spawn for all non-members of the spawn region

There are different flags for item pickup and item drop:

```
/rg flag spawn item-pickup -g nonmembers deny
/rg flag spawn item-drop -g nonmembers deny
```

Example: Blocking XP drops for all non-members of all plot regions

All the plot regions must be child of some template region. In this example, the template region is named `mall_parent`:

```
/rg flag mall_parent exp-drops -g nonmembers deny
```

All plots would inherit this flag and "nonmembers" would apply to non-members of each plot region.

This would **not work**:

```
DOESN'T WORK: /rg flag __global__ exp-drops -g nonmembers deny
```

This is because regions do not inherit from `__global__`, so "nonmembers" on `__global__` implies non-members of **specifically** the global region.

Blocks and Entities

One very important feature of how WorldGuard protects regions is how it handles blocks and entities. Players can obviously be either be added as a member of a protected region, but WorldGuard see blocks and entities the same way: they can either also be a member of a region.

However, entities and blocks cannot be explicitly added as member to a region. Instead, an entity or block is considered a member of a region if *it's within the region*. That's why, for example, a piston from outside a protected region cannot push into the region. It's because the piston is considered a non-member, and of course, someone or something that isn't a member cannot change blocks. On the other hand, a piston within a protected region can push blocks within the region because it's considered a member of the region.

WorldGuard also attempts to track the true *cause* of an event. For example, if a gravel block is placed above a protected region so that it falls into the protected region, WorldGuard considers the final placed block (that would exist after the gravel fell to the ground) to have been placed by the *falling block entity*, and the falling block entity to have been created by *the original gravel block high up*. (It's not as easy to determine who placed the original gravel block, however.) Because the original gravel block started outside the region, it cannot fall into the protected region because the original block was not a member.

Hint: When the `build` flag is set to `deny` on a region, no one can build and pistons don't work. That's because the build flag will even prevent members from building, as as detailed above, pistons can be regular members of a region like any other player.

Mod and Plugin Support

Some mods add new game mechanics, such as by adding new blocks, new items, new entities, or by adding new behavior to existing blocks, items, or entities. Mods be be written as Bukkit plugins or as mods for other modding platforms (such as Forge or LiteLoader).

Generally, Bukkit plugins tend to be better at respecting protection added by other Bukkit plugins, but this is not always the case. Non-Bukkit mods, however, tend to have extremely poor support. This section concerns mods that do not properly respect mods and plugins that aim to restrict access to an area, like WorldGuard.

Best-Effort Protection

What to watch out for

- Blocks or entities that change other blocks and entities
- Weapons, spells, or tools that shoot projectiles

WorldGuard is able to extend protection to most custom blocks and entities in most cases. Protection in this case is primarily handled by preventing the right click or left click of unknown blocks and entities on the server, which is usually sufficient as these are the only ways to interact with most blocks and entities.

However, WorldGuard cannot protect blocks or entities that open GUIs on the client (which you can usually tell on a multiplayer server if the GUI opens much quicker than inventories) because they send data in a separate channel that WorldGuard is not aware of.

In addition, WorldGuard inherently cannot effectively control actions (like dig blocks) on behalf of custom blocks or entities (such as a theoretical block mining drone), at least with high granularity. Base Minecraft itself does have blocks that can affect the world (like the piston), but the Bukkit team (or the maintainer of the server software that

you use) properly notifies plugins when base game blocks change the world. However, mod-added blocks and entities rarely do so, so WorldGuard is unable to deal with those cases.

Some mods "fake" a player in order to perform some actions on behalf of blocks and entities. The convention for these fake players is have their name be of the format `[ModName]`. However, this information is rarely useful beyond identifying which mod is making the change, which makes it impossible for WorldGuard to determine whether an action should be permitted or denied. To allow the mods to work at all in protected areas, fake players are given a free pass and can bypass all protection. This behavior can be disabled by setting `fake-player-build-override` in the *Configuration* but doing so prevents mods of this nature from working within a protected area.

Projectiles (and magic mods with projectile effects) are a major concern. This is because the mod likely does not notify plugins of effects of the projectile. WorldGuard has a workaround – the `emit-block-use-at-feet` setting in the *Configuration* that lets you configure a list of item types – that will *pretend* that listed items are trying to change the block at a player's feet. That effectively prevents a player *in* a protected region from firing his or her weapon because the player will be prevented from using the item in the region, but it does not prevent the player from firing into the protected region from *outside*.

Solutions

If the mod in question is a Bukkit plugin, please ask its developers to add support for protection plugins. That can be done by either firing appropriate Bukkit events or by using the *WorldGuard API*.

If the mod is not a Bukkit plugin, it is rare that you will be able to convince the developers to add support for Bukkit.

Alternatives to actually fixing protection are:

- To ignore the problem if you have a trustworthy userbase.
- To disable the items or blocks in question.

Common Scenarios

These are common scenarios that you may come across. Some of these scenarios have been already been described in some other pages.

- *General*
  - *How do I allow the usage of doors, levers, etc.?*
  - *How do I let people ride horses and Minecarts?*
  - *How do I prevent building in the wilderness?*
  - *How can I only deny exit for non-members?*
  - *How can I let players exit one side of an exit=deny region?*
  - *How do I unprotect all enchantment tables?*
  - *How do I allow mining but prevent block placing?*
  - *How do I allow players to read lectern books, but not take them?*
  - *How do I allow breaking of only certain block types?*
- *Plot Setups*
  - *How can I create a setup with plots?*

  - *How do I prevent door, etc. usage in public areas of the mall?*
- *Problems*
  - *Why don't pistons work?*

Tip: Be sure to also to check out the *Common Questions* page for more solutions to common problems.

General

How do I allow the usage of doors, levers, etc.?

If you want everyone to be able to use doors, levers, buttons, pressure plates, and so on, set the `use` flag:

```
/rg flag REGION_NAME use allow
```

If you want to apply it to **all** regions, set it on the *Global Region*:

```
/rg flag __global__ use allow
```

How do I let people ride horses and Minecarts?

If you want everyone to be able to use all vehicles, set the `ride` flag:

```
/rg flag REGION_NAME ride allow
```

If you want to apply it to **all** regions, set it on the *Global Region*:

```
/rg flag __global__ ride allow
```

Warning: This would allow players to take horses from others' regions! Only use this if you want that, or have other protections against stealing horses.

How do I prevent building in the wilderness?

As described on the *Global Region* page, you can set the `passthrough` flag to `deny`:

```
/rg flag __global__ passthrough deny
```

Do **not** set the `build` flag.

How can I only deny exit for non-members?

You want to set the region group of the flag (note that this is the default):

```
/rg addmember example_region sk89q
/rg flag example_region exit -g nonmembers deny
```

How can I let players exit one side of an exit=deny region?

Make two regions:

- One region would have `exit=deny`.
- The other region would border the side of the deny region, extending a bit outside, on the sides that you want to let players exit out of. This region would have `exit-overide=true`.

How do I unprotect all enchantment tables?

In the *Configuration*, adjust the `interaction-whitelist` setting and add the enchantment table. Get the proper names that you can use from Bukkit's Material.

How do I allow mining but prevent block placing?

Set the `block-break` flag to `allow`:

```
/rg flag mining_area block-break allow
```

How do I allow players to read lectern books, but not take them?

First, make sure you are on WorldGuard 7.0.1+. Earlier versions of WorldGuard released before 1.14 (and thus before lecterns) existed.

- Set the `use` flag to `allow`: `/rg flag <region> use allow`

How do I allow breaking of only certain block types?

Sorry, this is not supported per region yet! However, this can either be done per world in whitelist mode as described in *Blacklist* or using *Build Permissions*.

Plot Setups

How can I create a setup with plots?

If you want to create plots, you'd make use of region inheritence as described in *Priority and Inheritance*. There are two regions that you'd want to create, followed by the plot regions themselves.

You will have the following regions:

- The outer region (`mall`)
- An optional "template" region that all plot regions will inherit from (`shop_template`)
- The plots

Create the mall:

```
/rg define mall
```

Create the shop template. However, we'll use `-g` to make it a non-physical region since we're not actually using it to protect any area.

```
/rg define shop_template -g
```

Let's make some plots:

```
/rg define shop1
/rg define shop2
/rg define shop3
```

Now, you will need to set parents:

```
/rg setparent shop_template mall
/rg setparent shop1 shop_template
/rg setparent shop2 shop_template
/rg setparent shop3 shop_template
```

Hint: An alternative is to set the mall region to a lower priority (`/rg setpriority mall -1`).

How do I prevent door, etc. usage in public areas of the mall?

The goal here is:

- Set the `use` flag to `deny` in the mall: `/rg flag mall use deny`
- Since that prevents usage in the plots too, we need to change `use` within the plots only
- Since we created the template region, that makes it easy: `/rg flag shop_template use allow`

Due to the inheritance, the plots inherit the `use` flag from the template, which overrides the `use` flag on the mall.

Problems

Why don't pistons work?

Did you set the `build` flag? You probably do not want to do that. Be sure to remove it:

```
/rg flag __global__ build
```

- If you are trying to prevent building, you don't actually have to do anything! By default, when a region is made, it is protected.
- If you want to prevent building in the wilderness, you can set the `passthrough` flag to `deny`:

  ```
  /rg flag __global__ passthrough deny
  ```

Warning: At this time, it is not possible for a piston to push from one region into another. This issue is tracked as WORLDGUARD-3234 on the issue tracker.

1.7 Chest Protection

WorldGuard provides rudimentary self-serve chest protection that a player may utilize by placing a sign underneath their chest with specially formatted text. We generally advise new setups to **not** use this chest protection as it is not an actively updated part of WorldGuard. In addition, *regions* are a preferred method of defining ownership because it does not have problems involving blocks like hoppers.

> **Warning:** Chest protection in WorldGuard does not support UUIDs, so users will not be able to access chests if they change names. It also operates independently of regions and other protection.
>
> **It is highly recommended to not use this feature in WorldGuard, but use a dedicated plugin such as LWC Extended if you need single-block, sign-based protection. The feature will be removed from WorldGuard in a future version.**

1.7.1 Getting Started

Chest protection must first be enabled in the *Configuration*. With it disabled, chest protection will not be active but it will still not be possible to make "lock signs" (however, this can also be disabled in the configuration).

A sign is protected as long as a *special sign is placed under a chest*. This sign:

- Must be a sign post
- Must be under a chest or inventory (for double chests, it only needs to be under one side)
- Have the text `[Lock]` on the first line
- Have the player's username on the second line
- Have up to two more other players' names on the next two lines

It is not possible to place someone else's name on the second line.

1.8 WorldGuard API

You can access WorldGuard's data through its API, as documented here.

1.8.1 As a Dependency

In order to use WorldGuard's API in your plugin, you need to add WorldGuard as a dependency. It may be a required or optional dependency.

API Versions

Within major versions (5.x, 6.x, 7.x), WorldGuard's API is extremely stable. Deprecation tends to occur over a period of at least 3 months, but more typically over at 6 months.

The currently supported version of the API is 7.x. Older versions of the API (and of Minecraft) no longer receive updates or support. You can use the version selector in the bottom-right-hand corner of this page to access older versions of the documentation, but there may be changes or errors.

Build Script Dependency

If you compile your plugin or mod using something like Maven or Gradle (which you should!), you will need to add WorldGuard to the list of dependencies. You can find WorldGuard's artifacts in sk89q's Maven repository.

Note that some *objects that WorldGuard uses* come from WorldEdit. If you need to use those directly, you should also add WorldEdit as a compilation dependency (though it may be transitively provided depending on your build configuration).

- Maven repository: `https://maven.enginehub.org/repo/`
- Artifacts: `com.sk89q.worldguard:worldguard-bukkit:VERSION` (where VERSION is your desired version of WorldGuard); note that this contains the api in the `worldguard-core` artifact.

The Maven repository should be online 24/7, and is one of Minecraft's longest running Maven repositories. If it isn't, see *Getting Help*.

Example: Configuring a Maven pom.xml

```
<repositories>
    <repository>
        <id>sk89q-repo</id>
        <url>https://maven.enginehub.org/repo/</url>
    </repository>
</repositories>

<dependencies>
    <dependency>
        <groupId>com.sk89q.worldguard</groupId>
        <artifactId>worldguard-bukkit</artifactId>
        <version>VERSION</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Example: Configuring a Gradle build script

```
repositories {
    mavenCentral()
    maven { url "https://maven.enginehub.org/repo/" }
}

dependencies {
    compileOnly 'com.sk89q.worldguard:worldguard-bukkit:VERSION'
}
```

Modifying plugin.yml

First, it is important that you specify WorldGuard as a "depend" or "softdepend" in your plugin.yml file so that Bukkit knows to make sure that WorldGuard is loaded before your plugin:

```
name: My Plugin
version: 1.0
```

(continues on next page)

(continued from previous page)

```
description: This is my plugin!
depend: [WorldGuard]
```

If you choose to make it a soft dependency instead, WorldGuard will load first if it's installed, but otherwise your plugin will still load without WorldGuard.

Accessing WorldGuard From Your Plugin

Most WorldGuard APIs can be accessed using the `WorldGuard.getInstance()` method.

Adapting players can be done via the `wrapPlayer` method on `WorldGuardPlugin.inst()`. See *From Bukkit Objects* for details.

Note that you should keep *Internal APIs* in mind while accessing parts of WorldGuard.

Classpath Woes with Soft Dependencies

If you are using WorldGuard as hard dependency, you do not need to worry about WorldGuard classes potentially not existing at runtime. However, this is a concern if you are merely soft-depending on WorldGuard.

For example, if you tried to do:

```
class MyPlugin {
    public void onEnable() {
        ProtectedCuboidRegion region = new ProtectedCuboidRegion(...);
    }
}
```

Your plugin would not even load because `ProtectedCuboidRegion` could not be found. The plugin manager wouldn't even get to be able to call `onEnable()`. One way around this problem is to put the offending code in a different class entirely:

```
class RegionHolder {
    private final ProtectedCuboidRegion region;

    public RegionHolder() {
        region = new ProtectedCuboidRegion(...);
    }
}
```

While you have the same problem here where you can't even create a `RegionHolder`, at least you can catch the error from another class:

```
class MyPlugin {
    public void onEnable() {
        try {
            new RegionHolder();
        } catch (NoClassDefFoundError e) {
            // Do something here
        }
    }
}
```

However, this issue does not apply in all cases. Chained method calls, starting with a static method call, can be used without causing the containing class from failing to load:

```
class MyPlugin {
    public void onEnable() {
        try {
            boolean result = SomeClass.staticMethod();
        } catch (NoClassDefFoundError e) {
            // Do something here
        }
    }
}
```

It is therefore recommended that if you are using any sort of soft dependencies in your plugin, that you test your plugin without the soft dependencies installed.

1.8.2 Working with Regions

Region data can be accessed from other plugins using the region data API. A reference to WorldGuard should be available, which is explained in *As a Dependency*.

Regions and thread safety

The region API is completely thread safe. However, collections (lists, sets, maps) returned from various parts of the region API should generally not be modified. While many of these objects have been made immutable, there is some legacy code remaining.

Sections

Region data can be accessed via the *RegionContainer*. Every region is an instance of *ProtectedRegion*. If you want to check build permissions, check flags, find overlapping regions, or see what regions contain a point, you must perform a *spatial query*. Once you've gotten a spatial query result, you can *calculate flags* or *check build permissions*.

Managers

Region Container

Region data can be accessed via the `RegionContainer` object:

```
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
```

Every world has separate lists of regions. To access the regions for a particular world, the container has a `getWorld(World)` method:

```
RegionManager regions = container.get(world);
```

Tip: See *From Bukkit Objects* for how to convert Bukkit worlds.

Warning: The returned value **may be null** if region support is disabled or region data failed to load. WorldGuard may periodically attempt to load the data again.

Region Managers

To get a particular region by ID:

```
ProtectedRegion region = regions.getRegion("spawn");
```

In addition:

- Get an immutable map of all regions: `regions.getRegions()`
- Test whether a region exists given an identifier: `regions.hasRegion(String)`
- Get the number of regions: `regions.size()`

Example: Getting a region named "spawn"

```
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionManager regions = container.get(world);
if (regions != null) {
    return regions.getRegion("spawn");
} else {
    // The world has no region support or region data failed to load
}
```

Creating Regions

Once you've created an instance of a *ProtectedRegion*, use `addRegion(ProtectedRegion)` on a manager:

```
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionManager regions = container.get(world);
regions.addRegion(region);
```

Parent regions are automatically added. If there are existing regions with equivalent IDs, then the new regions will replace the previous regions.

Removing Regions

Regions can be removed by identifier using `regions.removeRegion(String, RemovalStrategy)`. The removal strategy parameter determines what to do regarding regions that inherit from the removed region.

```
regions.removeRegion("mall", RemovalStrategy.UNSET_PARENT_IN_CHILDREN);
```

Saving Changes

Region data is automatically saved after a short delay if changes are made, so there is **no need to explicitly save**.

If you wish to explicitly save, you can call either:

- `save()`
- `saveChanges()`

The calls can be used from any thread, but they will block until completion (or error).

Reloading Changes

To reload changes from disk, `load()` can be used. It can be called from any thread, but it will block until completion (or error).

Regions

Every region object is a subclass of the `ProtectedRegion` class, and there are several subclasses:

- `ProtectedCuboidRegion`
- `ProtectedPolygonalRegion`
- `GlobalProtectedRegion`

Each region object stores:

- An ID (immutable)
- Its priority
- Its (optional) parent
- A list of members
- A list of owners
- A list of region flags
- A boolean that automatically tracks whether the region has been modified

Vectors are used for referring to locations — these vector objects are from WorldEdit (see *From Bukkit Objects* for converting from Bukkit locations).

Example: Changing the priority of a region

```
region.setPriority(100);
```

Example: Setting the parent of a region

```
mall.setParent(null); // No parent
plot.setParent(mall);
```

If you attempt to create a scenario where there is circular inheritance, an exception will be thrown.

Example: Getting polygon vertices out of a region

```
if (region instanceof ProtectedPolygonalRegion) {
    ProtectedPolygonalRegion polygon = (ProtectedPolygonalRegion) region;
    List<BlockVector2D> points = polygon.getPoints();
}
```

Domains

Owners and members (`region.getOwners()` and `region.getMembers()`) are separate instances of `DefaultDomain`, which holds player names, player UUIDs, and permission groups.

Example: Adding members to a region

```
DefaultDomain members = region.getMembers();
members.addPlayer(UUID.fromString("0ea8eca3-dbf6-47cc-9d1a-c64551ca975c"));
members.addGroup("admins");
```

Warning: Referring to players by name (instead of by UUID) should generally not be done as names can be changed. Offline mode is not explicitly supported and those who use it do so at their own risk. Methods in Domains that use names instead of UUIDs are marked deprecated.

Example: Converting names to UUIDs in the background

If you need to convert player names into UUIDs, you should try to do it in the background if possible so that you do not pause the server or game.

You can use WorldGuard's `DomainInputResolver` class to help you do that. It implements `Callable<DefaultDomain>` and will return a `DefaultDomain` object that can be added to an existing domain. It takes input in the format of the member management commands. This is illustrated below.

```
// Google's Guava library provides useful concurrency classes.
// The following executor would be re-used in your plugin.
ListeningExecutorService executor =
        MoreExecutors.listeningDecorator(Executors.newCachedThreadPool());

String[] input = new String[] { "sk89q", "g:admins" };
ProfileService profiles = WorldGuard.getInstance().getProfileService();
DomainInputResolver resolver = new DomainInputResolver(profiles, input);
resolver.setLocatorPolicy(UserLocatorPolicy.UUID_AND_NAME);
ListenableFuture<DefaultDomain> future = executor.submit(resolver);

// Add a callback using Guava
Futures.addCallback(future, new FutureCallback<DefaultDomain>() {
    @Override
    public void onSuccess(DefaultDomain result) {
        region.getOwners().addAll(result);
    }

    @Override
    public void onFailure(Throwable throwable) {
        // Do something about the error
    }
});
```

It is highly recommended that you inform the user if the UUID lookup does not complete instantly.

Flags

Flags can be read by calling `getFlag(Flag flag)`. You can find static `Flag` objects on `Flags`:

```
Flags.BUILD
Flags.PVP
Flags.LEAF_DECAY
Flags.LIGHTNING
```

The returned value will be of the data type of the flag. For example, if you were to use `Flags.GREET_MESSAGE`, which is a `StringFlag`, a `String` will be returned.

Example: Getting the greeting message

```
String message = region.getFlag(Flags.GREET_MESSAGE);
player.sendMessage(message);
```

If the given flag is not set, `null` will be returned.

Setting Flags

Flags can be set using `setFlag(Flag flag, ? value)`. The value that you use must be of the type that the flag allows. For example, if the flag is a `StringFlag`, you can only set a `String`:

```
region.setFlag(Flags.GREET_MESSAGE, "Hi there!");
```

Flags can be removed by using `null` for the value.

Region groups can be set by calling `getRegionGroupFlag()` on a flag to get its region group flag:

```
RegionGroupFlag flag = Flags.PVP.getRegionGroupFlag();
```

Example: Setting the region group of the `use` flag:

```
region.setFlag(Flags.USE, StateFlag.State.ALLOW);
region.setFlag(Flags.USE.getRegionGroupFlag(), RegionGroup.MEMBERS);
```

Custom Flags

As of version 6.2, plugins can add their own flags and session handlers. See the *Custom Flags and Session Handlers* page.

Creating Regions

`ProtectedRegion` is an abstract class, so you must use one of the subclasses. For example, you might use a `ProtectedCuboidRegion`.

In every case, a region ID must be passed for the region. Valid regions must match the regular expression `^[A-Za-z0-9_,'\-\+/]{1,}` — that is, region IDs are only valid if they contain A-Z, a-z, 0-9, underscores, commas, single quotation marks, dashes, pluses, or forward slashes. IDs are case in-sensitive. The validity of an ID can be verified using `ProtectedRegion.isValidId(String)`.

To save a created region, see *Managers*.

Cuboids

To create a new cuboid region, two opposite corners are required. Any two opposite two corners are acceptable.

```
BlockVector3 min = BlockVector3.at(-10, 5, -4);
BlockVector3 max = BlockVector3.at(5, -8, 10);
ProtectedRegion region = new ProtectedCuboidRegion("spawn", min, max);
```

2D Polygon

Only 2D polygons are supported. These are polygons that have been extended vertically, which means that a minimum Y and a maximum Y are needed to create a 2D polygon region. A minimum of three points is required to create a valid 2D polygonal region.

```
List<BlockVector2> points = Lists.newArrayList(); // Call from Guava
points.add(BlockVector2.at(3, 4));
points.add(BlockVector2.at(0, 0));
points.add(BlockVector2.at(19, 3));
int minY = 0;
int maxY = 54;
ProtectedRegion region = new ProtectedPolygonalRegion("spawn", points, minY, maxY);
```

Global Regions

Not to be confused with *Global Region*, global regions have no physical area. They do not contain any points. The global region *does* use the `GlobalProtectedRegion`, but other regions can also utilize this class (users can create them by using the `-g` switch on `/rg define`).

These regions are usually used to create template regions for parenting.

```
ProtectedRegion region = new GlobalProtectedRegion("template");
```

Spatial Queries

There are a few methods to perform spatial queries on a specific region.

Hint: If you are interested in performing spatial queries on all regions, see *Spatial Queries*.

Testing Point Containment

`boolean contains(BlockVector3)` can be used to test whether a region contains a particular point.

Example: Seeing whether a region contains (20, 0, 30)

```
region.contains(BlockVector3.at(20, 0, 30));
```

Finding Intersecting Regions

The `getIntersectingRegions(Collection<ProtectedRegion>)` method call can be used to return a list of intersecting regions. These regions do **not** have to be fully contained.

Example: Seeing which regions overlap with spawn

```
List<ProtectedRegion> candidates = Lists.newArrayList();
candidates.add(mall);
candidates.add(hospital);

List<ProtectedRegion> overlapping = spawn.getIntersectingRegions(candidates);
```

Dirty Flag

Whenever changes are made to a region object, a "dirty" flag (not to be confused with region flags) is set on the region. This can be tested with `isDirty()`, and it is used by region managers to know which regions need to be saved.

Custom Flags and Session Handlers

As of WorldGuard 6.2, custom flags and session handlers are supported. This allows third-party plugins to create their own flags for WorldGuard regions.

Registering New Flags

Make sure that you have added WorldGuard as a *As a Dependency* first. The order of registration is very important, so WorldGuard must load before your plugin for this to work.

Flags have to be registered with WorldGuard's `FlagRegistry` with the `register(Flag<?> flag)` method. The parameter should be an instance of any flag object, whether you use one of the default types or your own type.

Registering has to be done before WorldGuard is **enabled**. Thus, it is highly recommended that you register when your plugin **loads**. After WorldGuard is enabled, the FlagRegistry is locked and no new flags can be registered.

Example: Registering a custom flag

```
// declare your flag as a field accessible to other parts of your code (so you can␣
↪use this to check it)
// note: if you want to use a different type of flag, make sure you change␣
↪StateFlag here and below to that type
public static StateFlag MY_CUSTOM_FLAG;

@Override
public void onLoad() {
    // ... do your own plugin things, etc

    FlagRegistry registry = WorldGuard.getInstance().getFlagRegistry();
    try {
        // create a flag with the name "my-custom-flag", defaulting to true
        StateFlag flag = new StateFlag("my-custom-flag", true);
        registry.register(flag);
        MY_CUSTOM_FLAG = flag; // only set our field if there was no error
    } catch (FlagConflictException e) {
        // some other plugin registered a flag by the same name already.
        // you can use the existing flag, but this may cause conflicts - be sure to␣
↪check type
        Flag<?> existing = registry.get("my-custom-flag");
        if (existing instanceof StateFlag) {
            MY_CUSTOM_FLAG = (StateFlag) existing;
        } else {
            // types don't match - this is bad news! some other plugin conflicts␣
↪with you
            // hopefully this never actually happens
        }
    }
}
```

Once your flag is registered, WorldGuard will take care of loading and saving it from the region database, allowing users to set it via the `/rg flag` commands, and so on. Even if your plugin is removed from the server, WorldGuard will keep the flag saved to any regions it was set on, but it will be rendered inert until your plugin is loaded again.

Using Session Handlers

In general, your flags will be used in event handlers by *querying values*. However, some flags may be designed for specific tasks such as:

- Running some method periodically on all players in regions with the given flag (e.g. heal flag)
- Responding to a player entering or leaving a region with the flag set (e.g. greeting flag)

Custom handlers can be registered any time after WorldGuard enables. A handler is instantiated by a factory method for each session (player) when it is created. There are also some methods in the Handler class which your custom handler has to override. If you are using a handler for the second type of behavior, it is recommended that you extend the FlagValueChangeHandler class instead, which handles a lot of the region-border-crossing logic for you.

Example: Creating and registering a custom handler

```
public class CustomHandler extends FlagValueChangeHandler<State> {
    public static final Factory FACTORY = new Factory();
    public static class Factory extends Handler.Factory<CustomHandler> {
        @Override
        public CustomHandler create(Session session) {
            // create an instance of a handler for the particular session
            // if you need to pass certain variables based on, for example, the␣
↪player
            // whose session this is, do it here
            return new CustomHandler(session);
        }
    }
    // construct with your desired flag to track changes
    public CustomHandler(Session session) {
        super(session, MyPlugin.MY_CUSTOM_FLAG);
    }
    // ... override handler methods here
}
```

```
SessionManager sessionManager = WorldGuard.getInstance().getPlatform().
↪getSessionManager();
 // second param allows for ordering of handlers - see the JavaDocs
sessionManager.registerHandler(MyCustomHandler.FACTORY, null);
```

Tip: WorldGuard's inbuilt handlers can be found in the com.sk89q.worldguard.session.handler package. These should serve as good examples for implementing your own handlers.

Note: Not all of WorldGuard's Session uses are flag-related. WorldGuard also uses Sessions to manage god mode, for example.

Spatial Queries

WorldGuard optimizes for two types of spatial queries:

- Finding all regions containing a point
- Finding regions overlapping another region

Spatial queries can be performed using an instance of a *RegionManager*, but they can also be performed through a special query cache. In either case, a returned `ApplicableRegionSet` object will be returned that contains a list of regions, as well as additional methods to perform *Flag Calculation*.

It's also possible to, given a list of regions, to create your own `ApplicableRegionSet`. This is useful if you want to *test the value of flags* and you have already a list of regions to test with (and the regions do not even need to overlap).

Getting an ApplicableRegionSet

Through the Query Cache

The query cache stores the last query result for at most one or two seconds, which speeds up repeated lookups for the same a block, a common scenario during event handling. However, the query cache only supports the

first type of query, point location. To utilize the cache, a new `RegionQuery` instance can be obtained from a `RegionContainer` (see *From Bukkit Objects* for converting from Bukkit locations):

```
RegionQuery query = container.createQuery();
ApplicableRegionSet set = query.getApplicableRegions(loc);
```

Example: Getting regions at (10, 64, 100)

```
Location loc = new Location(world, 10, 64, 100);
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionQuery query = container.createQuery();
ApplicableRegionSet set = query.getApplicableRegions(loc);
```

One particular feature of the cache is that it will **return virtual results** if region protection is disabled or region data failed to load:

- If region protection is disabled, an instance of `PermissiveRegionSet` will be returned, which will contain no regions and permit any action.
- If region data failed to load, an instance of `FailedLoadRegionSet` will be returned, which will contain no regions, deny every action, and also implicitly provide values for some flags (such as `deny-message` to inform players of the error).

In either case, `set.isVirtual()` will return `true` if the result is virtual.

Through a RegionManager

Given an *RegionManager*, `getApplicableRegions(Vector)` can be used to perform a point location query (see *From Bukkit Objects* for converting from Bukkit locations).

```
BlockVector3 position = BlockVector3.at(20, 10, 4);
ApplicableRegionSet set = regions.getApplicableRegions(position);
```

Example: Getting regions at (10, 64, 100)

```
Location loc = new Location(world, 10, 64, 100);
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionManager regions = container.get(world);
// Check to make sure that "regions" is not null
ApplicableRegionSet set = regions.getApplicableRegions(loc.toVector().
↪toBlockPoint());
```

If the goal is to find a list of regions that overlap another, use `getApplicableRegions(ProtectedRegion)` on the manager. Because shape and the region are currently the same objects (coupled together), you have to use a dummy ID:

```
BlockVector3 min = BlockVector3.at(0, 0, 0);
BlockVector3 max = BlockVector3.at(10, 10, 10);
ProtectedRegion test = new ProtectedCuboidRegion("dummy", min, max);
ApplicableRegionSet set = regions.getApplicableRegions(test);
```

Constructing Manually

`RegionResultSet` takes a `List<ProtectedRegion>` and an optional global region.

The provided regions do not need to overlap.

```
List<ProtectedRegion> regions = Lists.newArrayList();
regions.add(spawn);
regions.add(mall);
regions.add(pub);

ApplicableRegionSet set = new RegionResultSet(regions, null); // No global region
```

Warning: Your list of regions may be re-ordered in-place. After you have given a list of regions to the instance, it should no longer be used.

Using a ApplicableRegionSet

If your interest is in getting the list of regions, `ApplicableRegionSet` implements `Iterable<ProtectedRegion>` so you can loop over it:

```
for (ProtectedRegion region : set) {
    // Do something with each region
}
```

Example: Getting a list of regions

Google's Guava library has `Lists.newArrayList(Iterable)` to create an `ArrayList` from an `Iterable`.

```
List<ProtectedRegion> region = Lists.newArrayList(set);
```

If you are performing a spatial query to check protection or flags, see either *Querying Protection* or *Flag Calculation*.

Flag Calculation

To check flags, an instance of `ApplicableRegionSet` (which contains a list of regions) must have been retrieved. That is documented in *Spatial Queries*.

Querying Flags

One issue with query flags given a list of regions is that there may be several regions with the same flag assigned (and with different values). If priorities and child inheritance are properly configured, then there might only be one "effective" value, but otherwise there will be several.

Getting All Values

`queryAllValues(RegionAssociable, Flag)` can be used to get all the values that have been set for a flag. Flags can be obtained from `Flags`.

Example: Getting the greeting message flag, given a player

```
LocalPlayer localPlayer = WorldGuardPlugin.inst().wrapPlayer(player);
Collection<String> greetings = set.queryAllValues(localPlayer, Flags.GREET_MESSAGE);
```

Getting One Value

`queryValue(RegionAssociable, Flag)` can be used to get one value. Depending on the flag type, this may be the first value found, or it may be a "best" value. As of writing, only `StateFlags` will actually pick a "best" value.

Example: Getting the greeting message flag, given a player

```
LocalPlayer localPlayer = WorldGuardPlugin.inst().wrapPlayer(player);
String greeting = set.queryValue(localPlayer, Flags.GREET_MESSAGE);
```

The returned value may be `null` if the flag is not set on any regions.

Getting StateFlag Values

If you are trying to query a `StateFlag` (a flag with allow/deny), there are additional methods that you can use. They allow you to specify multiple state flags for automatic combination (remember, "deny" overrides "allow").

- `queryState(RegionAssociable, StateFlag...)` takes one or more flags and returns an allow, deny, or null
- `testState(RegionAssociable, StateFlag...)` does the same thing, but returns true if it's allow

You can still use `queryValue`, but you can only specify one flag at a time.

Example: Testing the build flag

```
LocalPlayer localPlayer = WorldGuardPlugin.inst().wrapPlayer(player);
if (!set.testState(localPlayer, Flags.BUILD)) {
    event.setCancelled(true);
}
```

Flags With No Player

If you are trying to lookup a flag that doesn't use a player (for example, the `creeper-explosion` flag), then you can use `null` for the `RegionAssociable`.

Example: Testing the creeper explosion flag

```
if (!set.testState(null, Flags.CREEPER_EXPLOSION)) {
    event.setCancelled(true);
}
```

Hint: You can also pass in `null` for flags that do use a player, but then *flag region groups* will not work correctly.

Also via RegionQuery

The methods described on this page are also conveniently available directly on instances of `RegionQuery`.

Example: Using a `RegionQuery` to directly query flags

```
LocalPlayer localPlayer = WorldGuardPlugin.inst().wrapPlayer(player);
Location loc = new Location(world, 10, 64, 100);
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionQuery query = container.createQuery();

// No need to bother:
// ApplicableRegionSet set = query.getApplicableRegions(loc);

// Just directly test the flag
query.testState(loc, localPlayer, Flags.BUILD);
```

In addition, you can use `testBuild` and so on as a shortcut to `testState(..., Flags.BUILD, your flags)`.

Non-Player Actors

Instead of passing in a player, you can instead pass in a (non-`LocalPlayer`) `RegionAssociable`. This object is used to determine whether to use rules for owners, members, or non-members should be used.

However, let's first consider what happens with players. Given a player part of the build team, who has been made an owner of both spawn's region and the "builder's club," the association returned should be `OWNER`, as illustrated below:

```
List<ProtectedRegion> regions = Arrays.asList(spawnRegion, buildersClub);
builderPlayer.getAssociation(regions) == Association.OWNER;
```

As you may be aware, you cannot add entities or blocks as members to a region, so it can't work the same way. To do that, a special `RegionAssociable` is used for blocks and entities: it takes a list of **source regions** to determine whether the source regions should be considered a "member" of the target location. This is illustrated below.

```
Set deepInside    = newHashSet(spawn, mall);
Set inside        = newHashSet(spawn);
Set outside       = newHashSet(); // Empty set

// outside -> inside = considered as "non-member"
new RegionOverlapAssociation(outside).getAssociation(inside) == NON_MEMBER

// inside -> inside = considered as "member"
new RegionOverlapAssociation(inside).getAssociation(inside) == OWNER

// inside -> deepInside = considered as "member"
```

(continues on next page)

(continued from previous page)

```
// Note that by default building is blocked in this case.
// The association must be at least "member" for all regions individually.
new RegionOverlapAssociation(inside).getAssociation(deepInside) == OWNER

// inside -> outside = considered as "non-member"
new RegionOverlapAssociation(inside).getAssociation(outside) == NON_MEMBER

// deepInside -> inside = considered as "member"
new RegionOverlapAssociation(deepInside).getAssociation(inside) == OWNER
```

Note that the `nonplayer-protection-domains` flag and region inheritance can override this behavior. The various `test...` and `query...` methods will handle this for you.

To summarize:

- Player (`LocalPlayer`) objects already implement `RegionAssociable`
- For entities and blocks, WorldGuard uses the regions where the block or entity is (`RegionOverlapAssociation`)

There is also:

- `ConstantAssociation` uses a pre-set type of association (`new ConstantAssociation(Association.MEMBER)` or `Associables.constant(Association.MEMBER)`)
- `DelayedRegionOverlapAssociation` which works like `RegionOverlapAssociation`, but doesn't do the spatial query for regions at the source until it is needed

Example: Examining how WorldGuard handles region protection

First, the correct `RegionAssociation` must be created for the event. `createRegionAssociable()` described below takes an object and returns a `RegionAssociable`.

```
private RegionAssociable createRegionAssociable(Object cause) {
    if (cause instanceof Player) {
        return WorldGuardPlugin.inst().wrapPlayer((Player) cause);
    } else if (cause instanceof Entity entity) {
        RegionQuery query = WorldGuard.getInstance().getPlatform().
↪getRegionContainer().createQuery();
        WorldConfiguration config = WorldGuard.getInstance().getPlatform().
↪getGlobalStateManager().get(BukkitAdapter.adapt(entity.getWorld()));
        Location loc = entity.getLocation(); // getOrigin() can be used on Paper if␣
↪present
        return new DelayedRegionOverlapAssociation(query, BukkitAdapter.adapt(loc),␣
↪config.useMaxPriorityAssociation);
    } else if (cause instanceof Block block) {
        RegionQuery query = WorldGuard.getInstance().getPlatform().
↪getRegionContainer().createQuery();
        WorldConfiguration config = WorldGuard.getInstance().getPlatform().
↪getGlobalStateManager().get(BukkitAdapter.adapt(block.getWorld()));
        Location loc = block.getLocation();
        return new DelayedRegionOverlapAssociation(query, BukkitAdapter.adapt(loc),␣
↪config.useMaxPriorityAssociation);
    } else {
        return Associables.constant(Association.NON_MEMBER);
    }
}
```

Let's see where it could be used:

```
@EventHandler
public void onPlayerBucketFill(PlayerBucketFillEvent event) {
    Player player = event.getPlayer();
    RegionAssociable associable = createRegionAssociable(player);

    if (!set.testState(associable, /* flags here */)) {
        event.setCancelled(true);
    }
}
```

Querying Protection

To query protection, the `Flags.BUILD` flag can be tested using the methods explained in *Flag Calculation*.

Warning: Region queries do not check if a player has bypass permissions. Depending on your use case, you may want to check that separately.

```
boolean canBypass = WorldGuard.getInstance().getPlatform().getSessionManager().
↪hasBypass(player, player.getWorld());
```

Example: Querying build permission using the query cache

```
LocalPlayer localPlayer = WorldGuardPlugin.inst().wrapPlayer(player);
Location loc = new Location(world, 10, 64, 100);
RegionContainer container = WorldGuard.getInstance().getPlatform().
↪getRegionContainer();
RegionQuery query = container.createQuery();

if (!query.testState(loc, localPlayer, Flags.BUILD)) {
    // Can't build
}
```

Region Events

WorldGuard currently does not fire very many events — in fact, only one at the moment.

Disallowed PVP

`DisallowedPVPEvent` is thrown when PvP is blocked by WorldGuard. It can be "unblocked" by cancelling the event.

1.8.3 From Bukkit Objects

Players

For legacy reasons and future cross-platform support, WorldGuard uses its own internal player object called `LocalPlayer`. A Bukkit `Player` can be converted to a `LocalPlayer` using `wrapPlayer(Player)` on `WorldGuardPlugin`.

```
WorldGuardPlugin.inst().wrapPlayer(player);
```

Other Types

Many operations need WorldEdit locations, worlds, and so on. Bukkit's `Location`, `World`, and other objects can be converted to WorldEdit objects using methods in `com.sk89q.worldedit.bukkit.BukkitAdapter`.

```
BukkitAdapter.adapt(location);
BukkitAdapter.adapt(world);
```

1.8.4 Internal APIs

Some of WorldGuard is not considered public API and may be changed at any moment without warning. Usage of this code is not considered proper, and will receive no support.

The precise definition of internal API is anything not accessible according to standard Java access rules, and any of the following types:

- Anything in the platform implementations (i.e. `worldguard-bukkit`, as opposed to `-core`) with the exception of the `wrapPlayer` method on `WorldGuardPlugin`
- **Anything in the following packages:**
  - `com.sk89q.worldguard.internal`
  - `com.sk89q.worldguard.commands`
  - `com.sk89q.worldguard.util`
- Anything explicitly marked as internal (in javadocs).

1.9 Advanced Topics

1.9.1 Event Logging

Event Funneling

Bukkit notifies plugins of things that happen in the game via "events." There are many, many events, such as:

- Bucket fill
- Bucket empty
- Right click of an entity by a player
- Placement of a block by a player
- Digging of a block by a player
- Change of a block by an entity

- Piston push
- Push retract

However, it really comes down to three fundamental objects in Minecraft:

- Items
- Blocks
- Entities

You can do certain things with these objects:

- Place them
- Break them
- Interact with them
- Damage them

To simplify matters, WorldGuard funnels the Bukkit events into it a more fundamental set of events:

- Bucket fill → Interact with a block, Interact with an item
- Bucket empty → Interact with a block, Interact with an item
- Right click of an entity by a player → Interact with an entity
- Placement of a block by a player → Interact with a block
- Digging of a block by a player → Interact with a block
- Change of a block by an entity → Interact with a block
- Piston push → Interact with a block
- Push retract → Interact with a block

Parts of WorldGuard, like region protection, only then need to handle "interact with a block," "interact with an entity," and so on and determine whether a block or entity can be placed, broken, or interacted with.

Tracking Cause

Another important facet of figuring out whether something should be permitted is to determine *who* is doing it.

What can make this complex is that causes are something indirect. For example, if a player shoots an arrow at another player, the immediate cause — the arrow — is not the true cause; rather, the player is. Another example is the act of placing a block of gravel in way so it falls down: the block that ends up on the ground is caused by (1) the falling gravel block entity, which was caused by the (2) original gravel block, which was originally caused by (3) a player.

Keep in mind that a cause could be a player, or it could be a block (pistons) or entity (Endermen or Creepers).

However, it is not always possible to track the true cause of an event. Sometimes WorldGuard must deal with untraceable chains of events.

Displaying Internal Events

There is only a handful of internal events that are used to funnel Bukkit's events. It's possible to output these events as they are created to the server log, which allows you to:

- Figure out which blacklist event is being triggered for certain actions
- See whether WorldGuard is handling an action at all

- Assist in development of WorldGuard to see how it handles certain events

To use this mode, specify `-Dworldguard.debug.listener=true` on the command line.

Tip: This feature should be enabled on a private test server as it will emit many log entries on a busy server.

Example: Enabling the mode in a Batch file on Windows

If your batch file reads like:

```
@ECHO OFF
SET BINDIR=%~dp0
CD /D "%BINDIR%"
"%ProgramFiles(x86)%\Java\jre7\bin\java.exe" -Xincgc -Xmx1G -jar craftbukkit.jar
PAUSE
```

You'd add `-Dworldguard.debug.listener=true` like so:

```
@ECHO OFF
SET BINDIR=%~dp0
CD /D "%BINDIR%"
"%ProgramFiles(x86)%\Java\jre7\bin\java.exe" -Dworldguard.debug.listener=true -
↪Xincgc -Xmx1G -jar craftbukkit.jar
PAUSE
```

The option can go anywhere after "java.exe" but before "-jar". In this case, the example puts it right after java.exe.

Interpreting the Output

Let's take the example of placing gravel above a protected region. You'd see, in your server log or console, the following output:

```
* USE   GRAVEL          [Player{sk89q}] @world :BlockPlaceEvent
* PLACE GRAVEL @0,99,0 [Player{sk89q}] :BlockPlaceEvent
* SPAWN FALLING_BLOCK  [Block{0,99,0}] @-0,99,0 :EntityChangeBlockEvent
* PLACE GRAVEL @       [Block{0,99,0} | FallingSand] :EntityChangeBlockEvent␣
↪[CANCELLED]
* SPAWN DROPPED_ITEM   [Block{0,99,0} | FallingSand] @-0,0,0 :EntityChangeBlockEvent
```

Note: The output has been been condensed and formatted for purposes of explanation.

The general syntax for each line is:

```
ACTION TYPE/LOCATION [CAUSES] @LOOCATION :BUKKIT-EVENT [CANCELLED?]
```

A cancelled event is one that has been blocked.

Tracing Gravel Placement

First, when the gravel block is used, it emits a use block event:

```
* USE    GRAVEL          [Player{sk89q}] @world :BlockPlaceEvent
```

The cause is, of course, the player. If the use block event isn't cancelled (due to the blacklist or some WorldGuard feature), then it moves onto the actual placement:

```
* PLACE GRAVEL @0,99,0 [Player{sk89q}] :BlockPlaceEvent
```

Because the gravel block is placed in the air, it must drop. Dropping blocks become entities (like skeletons or paintings), so this results in an entity spawn event with the cause being the placed gravel block. Note that WorldGuard does not attempt to track who placed the original gravel block.

```
* SPAWN FALLING_BLOCK  [Block{0,99,0}] @-0,99,0 :EntityChangeBlockEvent
```

When the (falling) gravel block hits the ground, it attempts to place a new gravel block on the ground and then removes itself, the falling block entity. The direct cause is the falling block entity, but the chain of events started with the placed block, and this is illustrated in the log entry:

```
* PLACE GRAVEL @        [Block{0,99,0} | FallingSand] :EntityChangeBlockEvent␣
↪[CANCELLED]
```

As you can see, the gravel placement event was blocked. This is because the gravel fell in a protected region and it originated from outside the region. Because WorldGuard doesn't want to cause players to lose their blocks accidentally on survival, an item is spawned instead, which results in another internal event:

```
* SPAWN DROPPED_ITEM   [Block{0,99,0} | FallingSand] @-0,0,0 :EntityChangeBlockEvent
```

You can see that the cause still extends all the way back to the original placed block. Because item drops for non-members (the original gravel block is a non-member because it came from outside the region) are not disabled in the region, the item event is not cancelled and the item drop is made.

1.10 Common Questions

- *General*
  - *Why don't any commands work?*
  - *How old is WorldGuard?*
  - *Who works on WorldGuard?*
- *Can't Build*
  - *Why can't players break or place blocks?*
  - *After setting up some regions, why can no one build?*
- *Building Not Blocked*
  - *Why is protection not working? Players receive NO message.*
  - *Why is protection not working? Players DO receive a message.*
- *Region Protection*
  - *Why do pistons not work?*
  - *How do I do _____ with region protection?*

Tip: Be sure to also to check out the *Common Scenarios* page for solutions to common issues regarding region protection.

1.10.1 General

Why don't any commands work?

If no commands work, it may because WorldGuard failed to start:

- Make sure that you are running Bukkit (or a server software that supports Bukkit). Use the `version` command in console or in-game and make sure that the response has "Bukkit" or "CraftBukkit" in it.
- Make sure that you have WorldEdit installed.
- Make sure that you have the proper version of WorldGuard for your version of Minecraft.

If those solutions do not help you, you will need to look through your startup log:

- If you use a game server host, use its log viewer.
- You can also open up "latest.log" in the logs folder of your server directory. (On older versions of Minecraft, the log file was "server.log" in the root directory.)

If you are unable to discover the problem from reading the server log, you can *ask for help or submit a bug report*.

How old is WorldGuard?

WorldGuard began in November 2010 for the "hMod" modding platform by sk89q. Later on, WorldGuard was ported to Bukkit.

Who works on WorldGuard?

WorldGuard has been developed by many people, and large portions of WorldGuard include contributed code. The list of top contributors can be found on GitHub.

1.10.2 Can't Build

Why can't players break or place blocks?

Note: A new WorldGuard installation has most features disabled, so it is unlikely to be caused by WorldGuard in such scenarios.

One of the easiest ways to identify the reason is to identify the message that you get when you are unable to build. WorldGuard usually uses dark red or crimson-colored messages, or a message like "Hey! Sorry, but you can't _____ here." If you get no message, it is probably not WorldGuard.

If it does not immediately appear to be WorldGuard,

- Make sure that Minecraft's "spawn protection" is not on. Spawn protection prevents anyone from making any changes within a certain radius of a world's spawn point. To adjust spawn protection, change the `spawn-protection` setting in the server.properties file to `0`.

- Make sure that you are not testing in adventure mode.
- Update your version of CraftBukkit/Spigot/Paper.

If those steps do not help, there's a simple command in WorldGuard that can simulate an action on behalf of a player and then report the plugin that blocked the action. Use the "testbreak" and "testplace" commands described on the *Commands* page to identify the plugin.

If the cause is WorldGuard:

- Use the *Region Wand* to see if any regions protect the blocks in question. If some do, but you do not know why, see *After setting up some regions, why can no one build?*
- Check to see whether *Build Permissions* is enabled.
- Check to see whether any other *Configuration* options are enabled that would prevent the action.
- If you cannot figure out the cause, *look into getting help*.

If the command lists a different plugin:

- See if you need to give extra permissions or change any configuration for the other plugin.

If the test command lists no plugins, make sure that you first followed the steps above (check spawn protection, etc.), then *look into getting help*.

After setting up some regions, why can no one build?

Use the *Region Wand* and right click an affected block to list the available regions. Use the `/rg info` to lookup information about each listed region.

- Make sure to verify membership of all the regions.
- Make sure that the `build` flag is not set to `deny`.

Is there only a global region?

- Make sure that the *Global Region* does not have `build` set to deny.
- Make sure that the *Global Region* does not have `passthrough` set to deny.
- Make sure that the *Global Region* does not have members or owners.

1.10.3 Building Not Blocked

Why is protection not working? Players receive NO message.

- You have op or full permissions, so you override protection.
- You have set the `build` flag on a region to "allow."
- You have set the `passthrough` flag on a region to "allow."
- You have set some other relevant flag (`pvp`, `ride`, etc.) on the region to "allow."
- The item in question is from a mod or a third-party plugin (see *What's Protected?*).
- WorldGuard does not yet protect that particular thing that you are trying to do. This is not the case for simple block place or break. Please make sure that you are using the latest version of WorldGuard, and if it's still a problem, *file a bug report*.
- There is a bug in your version of CraftBukkit/Spigot/Paper.

If you still cannot figure out the cause, *look into getting help*.

Why is protection not working? Players DO receive a message.

If WorldGuard is blocking an action, it's still possible for a different plugin to *unblock* the action. However, because WorldGuard has already sent the "you can't build" message, the player still receives it.

You can use the "testbreak" and "testplace" commands described in *Commands* to identify the causing plugin. If you see any plugin above WorldGuard on the list with "ALLOW" next to its line, then that plugin is the cause.

Another possibility is that your version of CraftBukkit/Spigot/Paper has a bug. Be sure to use the latest available version. If you still cannot figure out the cause, *look into getting help*.

1.10.4 Region Protection

Why do pistons not work?

You probably set the `build` flag to `deny` when you probably should not have. Check out the *Common Scenarios* page for more information.

How do I do _____ with region protection?

Check out the *Common Scenarios* page.

1.11 Getting Help

If you have a question or have errors,

- Join our Discord (preferred)

If you have a feature request or bug report,

- Submit it to our issue tracker

1.12 Source Code

You can find the source code to WorldGuard on GitHub.

WorldGuard is open source. Contributions must be licensed under the GNU Lesser General Public License v3.

CHAPTER 2

Links

- WorldGuard Homepage
- Downloads for Bukkit
- Experimental Builds
- Discord Server

CHAPTER 3

Indices and Tables

- genindex
- search