# Glowstone Documentation

***Release 2018.8.0***

**The Glowstone Project**

**Mar 17, 2019**

# About

Welcome to the Glowstone documentation!

Glowstone is a fast, customizable and compatible open source Minecraft server written in Java that supports plugins for Bukkit, as continued by Spigot and Paper. Glowstone's code is completely from scratch, so some Bukkit plugins that use Mojang code will not work, though.

This documentation introduces Glowstone for server owners and developers. If you are someone that plays on a server powered by Glowstone, you may find our website more interesting.

If you need help with something or if the documentation can't answer your question, you can chat with us via our Discord server.

# CHAPTER 1

## Why Glowstone

**Glowstone is completely open-source**. We do not depend on internal Minecraft code, all of it is original. This also means our code is a lot simpler and much more optimized in most places than CraftBukkit and Minecraft.

Just because Glowstone is written from scratch does not mean you'll sacrifice plugin compatibility. Glowstone supports Bukkit, Spigot-API, and Paper-API plugins natively.

If you are a developer, you will find contributing a lot easier than with CraftBukkit, as there is no obfuscated code or a need to maintain a minimal diff. You could even modify Glowstone for private use, to strip away or enhance parts of Glowstone to your server's specific needs.

---

**Note:** We would like to say that Glowstone is not complete, and you will notice some missing features compared to your CraftBukkit-based server.

We also do not support "*NMS*" or "*OBC*" code, although enhanced compatibility is being worked on through Linkstone. In other words, Glowstone does not provide `org.bukkit.craftbukkit.*` and `net.minecraft.server.*` packages, so **plugins using these packages will break**.

---

# CHAPTER 2

## Install Java

Install **'OpenJDK 8'_** (recommended). If you're running a Linux distro, look into installing OpenJDK through your package manager before using these downloads.

---

**Note: We require Java 8 update 101 or greater.**

Our Maven repository uses a Let's Encrypt HTTPS certificate, and Let's Encrypt was added to the Java truststore in update 101.

Without this truststore, our library manager fails to pull dependencies from our server, due to an SSL security error.

---

Confirm that you have Java 8 by running `java -version` in your system's command prompt/terminal. You should see something like `java version "1.8.0_121"` at the top. As long as it includes `1.8`, and the number after the _ is more than 101, you're good.

---

**Note:** If you installed Java 8 on **macOS**, but `java -version` is showing an older version, run

```
sudo mv /usr/bin/java /usr/bin/java-1.6
sudo ln -s '/Library/Internet Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java
↪' /usr/bin/java
```

For more details, see this guide.

---

CHAPTER 3

# Install Glowstone

1. Download the latest build of Glowstone.
2. Move the jar file to your server folder. If you drop and run the jar in your current server folder, it will automatically migrate most settings to Glowstone.
3. Follow the instructions for your operating system below.

## 3.1 Windows

Using a text editor like Notepad, create a new start script named `start.bat` to launch the jar file:

```
@echo off
java -Xms768M -XX:+UseG1GC -jar glowstone.jar
pause
```

Make sure you select `All Files` as the file type in the save window.

Finally, double click the `start.bat` file.

## 3.2 GNU/Linux

Using a text editor like gedit, mousepad, Atom, Sublime Text, nano, (Neo)vi(m), Kate or emacs, create a new start script named `start.sh` to launch the jar file:

```
#!/bin/sh
BINDIR=$(dirname "$(readlink -fn "$0")")
cd "$BINDIR"
java -Xms768M -XX:+UseG1GC -jar glowstone.jar
```

Open terminal, go to your Glowstone folder and enter this command to give the script execute permissions:

```
chmod +x start.sh
```

Enter this command in the terminal to start the server:

```
./start.sh
```

## 3.3 macOS

Using a text editor like TextEdit, create a new start script named `start.command` to launch the jar file:

```
#!/bin/bash
cd '$( dirname "$0" )'
java -Xms768M -XX:+UseG1GC -jar glowstone.jar
```

Open terminal, change current directory by typing `cd` and dragging and dropping the Glowstone folder into terminal window.

At this point it should look like this:

```
cd /Users/YourName/YourGlowstoneFolder/Glowstone
```

If you think that is correct press enter and move to the next step.

Just **type** (do not enter!) this command to give the script execute permissions:

```
chmod a+x
```

Drag start.command into the Terminal window. Confirm there is a space between `chmod a+x` and the `start.command` path, and then enter. Finally, double click the `start.command` file to start the server.

# CHAPTER 4

# First Run

After having installed Glowstone for your operating system, you can start your server using the script you created.

**Tip:** If you want the server to simply generate/validate configuration files without starting the server, you can use the `--generate-config` command-line argument.

Simply add the argument at the end of the `java [...] -jar glowstone.jar` line inside your startup script.

## 4.1 Console Output

### 4.1.1 Server Version

Whenever you start the server, the first line will output the server version. For example:

```
[INFO] This server is running Glowstone version 2018.4.0-SNAPSHOT.be008ff (MC: 1.12.
↪2) (Implementing API version 1.12.2-R2.1-SNAPSHOT)
```

- The first information is the software version, `2018.4.0-SNAPSHOT`. The `SNAPSHOT` suffix is only present on development versions (i.e. not releases).
- The second part is a hash segment uniquely identifying the build you are using (`be008ff`).
- In parentheses is the Minecraft version supported by the server (`MC: 1.12.2`).
- Finally, the API version is the Glowkit version the server implements (`1.12.2-R2.1-SNAPSHOT`).

### 4.1.2 Library Downloads

When you first start the server, it will need to download some additional libraries from our repository in order to keep compatibility with some plugins. This may take a few seconds depending on your internet connection.

```
[INFO] Downloading org.apache.commons:commons-lang3:3.5...
[INFO] Downloaded org.apache.commons:commons-lang3:3.5.
       [ etc. ]
```

More information about libraries can be found in the **'Library Management'_** section.

### 4.1.3 Plugin Scanning

Your server will then scan for plugins inside the `plugins` directory. Once it finds compatible plugins, they will be loaded before the worlds are loaded. If you are familiar with Bukkit plugin development, this is the time when the `JavaPlugin#onLoad()` method is executed.

### 4.1.4 World Generation/Loading

The server will load world files from disk, located by default in the `worlds` directory. If it can't find a world, it will create it and start generating the terrain. Again, this may take a while depending on your hardware.

```
[INFO] Preparing spawn for world...
[INFO] Preparing spawn for world: 0%
[INFO] Preparing spawn for world: 2%
[INFO] Preparing spawn for world: 6%
       [ etc. ]
[INFO] Preparing spawn for world: done
```

### 4.1.5 Plugin Enabling

After the worlds are loaded and ready, the plugins that were previously loaded are now "enabled".

### 4.1.6 Server Binding

Once it is ready, the server will open itself on a TCP port. By default, this is port `25565`, and can be changed in the server configuration. For more information, refer to the *Configuration Guide* section.

```
[INFO] Binding server to 0.0.0.0:25565...
[INFO] Successfully bound server to 0.0.0.0:25565.
[INFO] Ready for connections.
```

Once the "Ready for connections." line is output, your server should be reachable by clients.

**Error: Failed to bind to address. Maybe it is already in use?**

This error means that you've already got a server running on the port that you've configured.

When starting the server, please make sure you do so by following the methods in the installation instructions, instead of double-clicking the JAR. If you do double-click the JAR, the server will start up, but you won't have any console, so you'll have to kill it using the task manager or whatever process management tools are relevant to your system.

- If you're converting from Bukkit, make sure you've stopped the old Bukkit server, if you plan to use the same port.
  - Make sure you're not running more than one copy of Glowstone on the same port.

- Check that you have no extra Java processes running. If you're on Windows, use the Task Manager. You might not want to kill some of them, such as the Minecraft client.
- If this still isn't working, please check whether you have an `ip` set in the `server` section of your `config/glowstone.yml` file.
  - If you do, please consider that most people will not need this entry - it's only useful for people running large servers with multiple network cards. In the majority of cases, you can simply remove this line from your file.
  - If you're convinced that you need it, it should contain only an IP address - usually of the form `a.b.c.d`, from `0.0.0.0` up to `255.255.255.255` for IPv4.

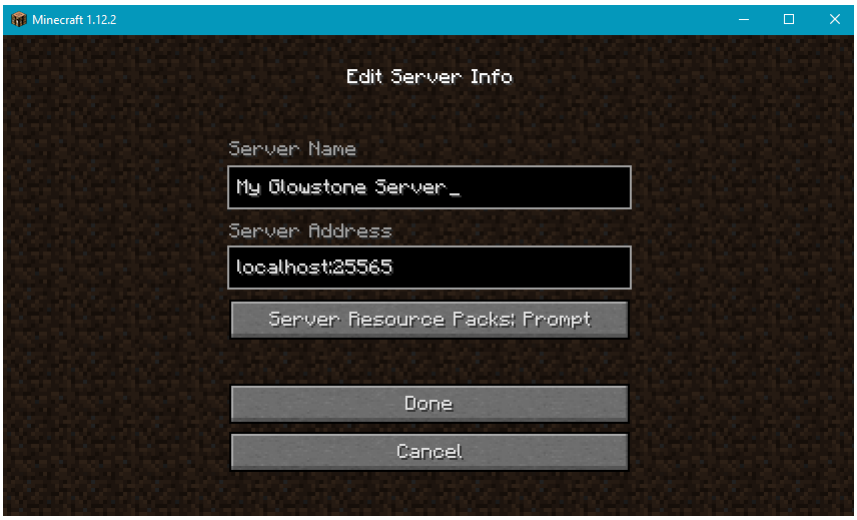
CHAPTER 5

# Connect to the Server

Once your server is open on a port, you can connect to it using a Minecraft client.

For the sake of demonstration, we will assume you are running your server locally on port `25565`.

1. Click on the **Multiplayer** button in the Minecraft main menu.
2. Click on **Add Server**.
3. Fill in a server name, and then set the **Server Address** to `localhost:25565`. It should look something like this:




4. Click on **Done**. Your server should now appear in your server list.

> **Note:** Sometimes, your server will show as unavailable ("*Can't connect to server*") after you add it to the server list.
>
> If that is the case, click on **Refresh**, and it should be live.

5. Select your server in the list, and click on **Join Server**. After a few seconds, you should spawn in your server's world.

CHAPTER 6

# Basic Administration

Administration tasks are done through commands, either inside the console (the terminal) or by players (in-game). These commands are used in the form `<command> [arguments...]`.

When describing a command, words inside angled brackets (`<...>`) represent **required** arguments, while square brackets (`[...]`) represent **optional** arguments.

---

**Note:** When using commands in-game, the forward-slash prefix (`/`) is added.

For example, a player would use `/kick otherplayer You are evil`, while the console operator would use `kick otherplayer You are evil`.

For the sake of consistency, the prefix will not be used in this documentation.

---

## 6.1 Operators

In order to give permission to a player to use administration commands, they will need to be made **operators** of the server.

A player can be added as a server operator using the `op <player>` command. For example, to add a player named "glowstoner" as an operator, use the command `op glowstoner`.

Similarly, to remove a player as an operator, use `deop <player>`. For example, `deop glowstoner`.

Players that attempt to use administrative commands without operator privileges will receive an error.

## 6.2 Kicking and Banning Players

In order to kick a player from the server, use the `kick <player> [reason]` command. The first argument is the name of the target player, and is required. The remainder of the command is optional and will be shown to the kicked player.

For example, `kick glowstoner Not today!` would show "Not today!" on *glowstoner*'s client.

Kicking a player is considered a "temporary" punishment, because the player can still log back into the server after being disconnected. In order to **ban** a player from joining the server, use the similar `ban <player> [reason]` command.

In order to un-ban a player, use the `pardon <player>` command.

## 6.3 Whitelisting

Whitelisting allows you to limit which players can join the server. When enabled, players not on the whitelist will not be able to join the server.

In order to enable the whitelist, use the `whitelist on` command. Similarly, use `whitelist off` to disable it.

To add a player to the whitelist, use `whitelist add <player>`. Oppositely, use `whitelist remove <player>` to remove them.

Operators can list players currently inside the whitelist using `whitelist list`.

CHAPTER 7

# Plugins

Most Bukkit, Spigot and Paper plugins will work on Glowstone.

As you would with a CraftBukkit server, place the plugin `.jar` files inside the `plugins` directory.

> **Warning:** Plugins that use internals for Paper, Spigot, CraftBukkit, or Minecraft will not work, unless they are designed to fail gracefully.
>
> This includes plugins that use "*NMS*" reflectively.

CHAPTER 8

# Additional Help

If you have any trouble setting up, we would be happy to help you on the forums or Discord.

- If you can't access your server from your public IP, make sure you have port forwarded or used a UPnP port mapper to allow incoming connections to your server.
- For server issues or suggestions, create an issue on Github.
- By default, all configuration files are in the `config` folder.

CHAPTER 9

# Configuration Guide

The YAML files allow you to edit your server settings. They can be found in the `config` directory in your server installation.

After modifying your configuration files, you will need to restart your server to apply the changes.

---

**Tip:** If you want the server to simply generate/validate configuration files without starting the server, you can use the `--generate-config` command-line argument.

Simply add the argument at the end of the `java [...] -jar glowstone.jar` line inside your startup script.

---

CHAPTER 10

# glowstone.yml

The `glowstone.yml` file allows you to edit your server settings.

This section documents the different options inside the file. We recommend using the navigation tree on the left to move around.

## 10.1 server

Basic server settings.

| Key | Type | Default | Description |
|---|---|---|---|
| ip | text *(optional)* | *(Blank)* | Which interface the server should listen on, usually blank. |
| port | integer | 25565 | The port the server should listen on. |
| name | text | Glowstone Server | The server's name, used in queries. |
| log-file | text | logs/log-%D.txt | Where the log is stored relative to the server folder. |
| online-mode | true/false | true | Whether connecting players are authenticated. Only disable this if you know what you are doing. |
| max-players | integer | 20 | The maximum number of players on the server. |
| whitelisted | true/false | false | Whether the whitelist is enabled. |
| motd | text | Glowstone Server | The message shown in the server list. |
| shutdown-message | text | Server shutting down. | The message used to kick players when the server stops. |
| allow-client-mods | true/false | true | Tell Forge clients whether or not the server allows for client mods. |
| snooper-enabled | true/false | false | Whether Minecraft stats reporting is enabled. Currently not implemented. |
| prevent-proxy-connections | true/false | true | Whether the server should verify that the IP that is used for connecting to the server is also used for authenticating with the Mojang servers. |

## 10.2 console

Settings for the console (terminal) output.

| Key | Type | Default | Description |
|---|---|---|---|
| use-jline | true/false | true | Whether the fancy console is enabled. Disable if you're having console problems. |
| prompt | text | '> ' | The console prompt that appears before the input field. |
| date-format | text | HH:mm:ss | How the time and date are displayed within the console. |
| log-date-format | text | yyyy/MM/dd HH:mm:ss | How the time and date are logged in the log files. |

## 10.3 game

Settings for in-game features.

| Key | Type | Default | Description |
|---|---|---|---|
| gamemode | text *(enum)* | SURVIVAL | The default gamemode, one of SURVIVAL, CREATIVE, ADVENTURE, or SPECTATOR. |
| gamemode-force | true/false | false | Whether players are forced to the default gamemode on join. |
| difficulty | text *(enum)* | NORMAL | The difficulty, one of PEACEFUL, EASY, NORMAL, or HARD. |
| hardcore | true/false | false | Whether hardcore mode (ban on death) is enabled. |
| pvp | true/false | true | Whether player vs. player mode is enabled. |
| max-build-height | integer | 256 | The maximum height at which players may build. |
| announce-achievements *(deprecated)* | true/false | true | Whether achievements are announced in the chat. Unused. |
| allow-flight | true/false | false | Whether unauthorized flight prevention is disabled. |
| command-blocks | true/false | false | Whether command blocks are enabled. Unused. |
| resource-pack | text *(optional)* | *(Blank)* | The URL of the resource pack to send to clients by default. |
| resource-pack-hash | text *(optional)* | *(Blank)* | The hash of the resource pack for data integrity purposes. |

## 10.4 creatures

Used to control mob spawn limits.

**Note:** Certain creature settings are not currently implemented, and are therefore not documented.

| Key | | Type | Default | Description |
|---|---|---|---|---|
| enable | monsters | true/false | true | Whether monsters can spawn naturally. |
| | animals | true/false | true | Whether animals can spawn naturally. |

## 10.5 folders

Settings for server folder names.

| Key | Type | Default | Description |
|---|---|---|---|
| plugins | text | plugins | The plugins directory relative to server root. |
| update | text | update | The directory relative to 'plugins' to copy updates from on startup. |
| worlds | text | worlds | The world container relative to server root. |
| libraries | text | lib | The libraries directory relative to server root. |

## 10.6 files

Settings for server file names. **These files are relative to the config directory.**

| Key | Type | Default | Description |
|---|---|---|---|
| permissions | text | permissions.yml | The file to read custom permissions from. |
| commands | text | commands.yml | The file to read command aliases from. |
| help | text | help.yml | The file to read help topics from. |

## 10.7 advanced

Advanced server configuration options.

| Key | | Type | Default | Description |
|---|---|---|---|---|
| connection-throttle | | integer | 4000 | Time in milliseconds a client must wait before reconnecting. |
| idle-timeout | | integer | 0 | How long until an idle (AFK) player is kicked (0 for never). |
| warn-on-overload | | true/false | true | Whether to show warnings if the server is overloaded. |
| exact-login-location | | true/false | false | Whether to skip fixing block collisions on player login. |
| plugin-profiling | | true/false | false | Whether the `timings` command is enabled. |
| deprecated-verbose | | true/false/default | default | Whether to always, never, or only sometimes show deprecation warnings for plugins. |
| compression-threshold | | integer | 256 | The minimum packet size to compress. -1 to disable, 0 to compress everything. |
| proxy-support | | true/false | false | Whether proxy (e.g. BungeeCord) support is enabled, granting access to the real IP and UUID of proxied players. Requires the proxy to be configured correctly. |
| player-sample-count | | integer | 12 | How many online players can be shown in the server list. |
| graphics-compute | enable | true/false | false | Whether GPU-based computations are enabled. |
| | use-any-device | true/false | false | Whether any device can be used for OpenCL computations. |
| region-file | cache-size | integer | 256 | The region file cache size, in MB. |
| | compression | true/false | true | Whether region files should be compressed. |
| profile-lookup-timeout | | integer | 5 | Timeout for Mojang profile lookups, in seconds. |
| suggest-player-name-when-null-tab-completions | | true/false | true | Checks if player names should be suggested when a command returns null as their tab completion result. |

## 10.8 extras

Extra services which Glowstone can optionally provide.

| Key | Type | Default | Description |
|---|---|---|---|
| query-enabled | true/false | false | Whether the query server is enabled. |
| query-port | integer | 25614 | The port the query server runs on. |
| query-plugins | true/false | true | Whether the query response includes plugin info. |
| rcon-enabled | true/false | false | Whether the rcon server is enabled. |
| rcon-password | text | glowstone | The rcon password. |
| rcon-port | integer | 25575 | The port the rcon server runs on. |
| rcon-colors | true/false | true | Whether the server should send color-codes to the rcon client. |

## 10.9 world

Used to choose how the default worlds are configured. For advanced world configuration, a plugin such as Multiverse may be appropriate.

| Key | Type | Default | Description |
|---|---|---|---|
| name | text | world | The name of the main world. |
| seed | text *(optional)* | *(Blank)* | The seed to use for the main world, or blank for random. |
| level-type | text *(enum)* | DEFAULT | The world type to use for the main world, one of DEFAULT, FLAT, DEFAULT_1_1, LARGEBIOMES, or AMPLIFIED. |
| spawn-radius | integer | 16 | The radius around a world's spawn point to protect from damage, or 0 to disable. |
| view-distance | integer | 8 | The radius of the area of chunks to send to players. |
| gen-structures | true/false | true | Whether structures (villages, strongholds, etc.) are generated. |
| allow-nether | true/false | true | Whether a Nether world is created by default. |
| allow-end | true/false | true | Whether an End world is created by default. |
| keep-spawn-loaded | true/false | true | Whether chunks around world spawns are kept loaded by default. |
| populate-anchored-chunks | true/false | false | Whether anchored chunks, like world spawns, are populated as soon as they are loaded. False means that these chunks will wait to be populated until a player loads those chunks for the first time, resulting in a long "Downloading terrain" wait time, and server stutter on first world join. |
| classic-style-water | true/false | false | Changes the water flow behavior to be finite with a moving source. |
| disable-generation | true/false | false | Disables world generation. |

## 10.10 libraries

| Key | Type | Default | Description |
|---|---|---|---|
| checksum-validation | true/false | true | Whether downloaded libraries should be validated using their checksum. |
| repository-url | text | *Glowstone repo, see below* | The repository URL to download libraries from. |
| download-attempts | integer | 2 | The maximum amount of attempts to download each library. |
| compatibility-bundle | text *(enum)* | CRAFTBUKKIT | The compatibility bundle to use. Only CRAFTBUKKIT and NONE are supported. See the **'Library Management'_** section. |
| list | list *(of libraries)* | *Empty array* | A list of extra libraries to download on server startup. See below for the content structure of this list. |

**Note**: the default Glowstone library repository is `https://repo.glowstone.net/service/local/repositories/central/content/`.

### 10.10.1 Library format

Each element of the library list (`libraries.list` key) is an object with the following structure:

| Key | | Type | Description |
|---|---|---|---|
| group-id | | text | The group ID of the library. |
| artifact-id | | text | The artifact ID of the library. |
| version | | text | The version of the library. |
| repository | | text *(optional)* | If present, overrides the default repository URL. |
| checksum *(optional)* | type | text *(enum)* | The algorithm for the checksum. Only SHA-1 (`sha1`) and MD5 (`md5`) are supported. |
| | value | text | The checksum of the library. |

Again, more information about these fields and library management can be found in the **'Library Management'_** section.

CHAPTER 11

# Errors

## 11.1 Failed to bind to address. Maybe it is already in use?

```
[SEVERE] Error during server startup.
java.lang.RuntimeException: Failed to bind to address. Maybe it is already in use?
        at net.glowstone.GlowServer.bind(GlowServer.java:438)
        at net.glowstone.GlowServer.main(GlowServer.java:93)
```

**This error means that you've already got a server running on the port that you've configured.**

More information can be found in the *Server Binding* subsection inside the First Run section.

## 11.2 Unsupported major.minor version

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: net/glowstone/
↪GlowServer : Unsupported major.minor version 52.0
```

**This error means that you're running a Java version older than Java 8, and you'll need to update.** On Windows and Linux, this should be fairly self-explanatory, but macOS users may find that they've already upgraded Java. If you use macOS, then you should follow this excellent guide.

## 11.3 java.lang.ClassNotFoundException: net.minecraft.server or org.bukkit.craftbukkit

```
[SEVERE] Could not load 'plugins/MassiveCore.jar' in folder 'plugins'
    org.bukkit.plugin.InvalidPluginException: java.lang.NoClassDefFoundError: net/
↪minecraft/server/v1_7_R4/PlayerInventory
```

This error is caused by plugins whose developers refused or were unable to use pure Bukkit classes when writing their plugins, and instead used CraftBukkit or internal Minecraft classes to achieve the functionality they needed.

These plugins are often called "impure" plugins, and aren't technically Bukkit plugins at all. **They will not work with Glowstone.** You will have to ask the developer of the plugin to add Glowstone support, or fix the plugin yourself, if you're able to.

Plugins that rely on ProtocolLib will also have the same issue, as it uses internal Minecraft classes.

CHAPTER 12

# Frequently Asked Features

Some features are frequently requested to us, and some will likely never be added officially.

## 12.1 Support for older versions

**We have taken the decision not to support older versions of Minecraft.**

We will always try to update to the latest Minecraft version once the Bukkit API is updated by the Spigot and Paper teams. However, you may be able to achieve this by using third-party tools and plugins like BungeeCord. We have also worked with ProtocolSupport's team to make it possible to host older Minecraft versions on the latest version of Glowstone.

## 12.2 Support for Sponge plugins

In the past, we have worked with the Sponge team to support the Sponge API on Glowstone. Unfortunately, we have dropped this objective and we will likely never officially support Sponge plugins on our platform. More details on this topic can be found here.

## 12.3 Support for Minecraft: Bedrock Edition

At this time, we simply do not have the time and resources to work on a Bedrock Edition port for Glowstone, and we would rather focus our work on the PC version. Some third-party projects (like ProtocolSupport) are currently working on this kind of support on Bukkit platforms.

## 12.4 Packet API / ProtocolLib support

Because networking is not part of the scope of the Bukkit API, there is no Packet API implementation in Glowstone. We are faced with a decision between making our own API, or supporting an API designed by the community. ProtocolLib has been around for quite a while and has become the *de-facto* community standard API for packet handling. ProtocolLib is designed to work on multiple versions of the Vanilla server using reflection, but it does not support Glowstone networking internals. This feature has been discussed on Github, but it may take a while before anything is functional.

# CHAPTER 13

# Inconsistencies

## 13.1 [Insert feature here] doesn't work!

Glowstone is a completely custom server software - that means that it does not use any of the Minecraft code. CraftBukkit (commonly known as Bukkit) heavily used internal Minecraft code for a lot of the things that it does.

As Glowstone doesn't, and can't, use the relevant Minecraft code for anything it implements, all of the features you'd expect from a Minecraft server must be rewritten from scratch.

As a result, Glowstone development is a little slow for some people's tastes, and leaves many new users somewhat confused. While we are working to be a complete Minecraft server replacement, these things take time.

## 13.2 I used the same world seed on [Platform], but Glowstone's world looks different

Because our terrain generator was made from scratch, using the same seed with another platform will not generate the same terrain (and may actually be completely different!). This is an expected caveat from a from-scratch implementation, because of how pseudo-randomization works.

CHAPTER 14

# Legal

**Warning: We are not lawyers, we're programmers.**

You should use the information pertaining to this section as a guideline, rather than a basis for your legal arguments in court.

## 14.1 If I use Glowstone, do I still have to follow the EULA?

**Yes, you still have to follow the Minecraft EULA.**

Glowstone is a Minecraft server - that is to say, it implements the Minecraft protocol, is designed to work with Minecraft clients, and allows users to play Minecraft with other users. Anyone that plays Minecraft in any capacity or runs a service that works with Minecraft players is required to follow the Minecraft EULA. See *Blizzard v bnetd*.

If you feel that this is incorrect, we suggest that you seek professional legal counsel before inadvertently breaking the EULA.

CHAPTER 15

# Contributing to Docs

This is a guide on how to setup your environment to write docs for this website.

We really appreciate users and developers contributing to our documentation!

## 15.1 Setup

### 15.1.1 System Requirements

In order to edit and build docs locally, you will need to have the following:

- Git
- Python 3.6.x
- Pipenv
- **Linux and Mac**: The `make` CLI

If you've never used Pipenv before, it is a useful tool to create Python virtual environments. You can get it using `pip` for your Python 3.6 environment:

```
python3.6 -m pip install pipenv
```

### 15.1.2 Local Docs Setup

First, create an account on GitHub, and fork the Glowstone docs repository to your account.

Then, clone your fork locally using:

```
git clone https://github.com/[your_username]/docs.git Glowstone-Docs
```

Switch to the directory that was created (`cd Glowstone-Docs`), then run the following to install the required dependencies:

```
pipenv sync --dev
```

Finally, create a branch for your changes. The name of the branch can describe what your changes are about.

```
git checkout -b my-branch-name
```

### 15.1.3 Editing Docs

Once your environment is setup, you can now modify the RST files inside of the repository.

To build docs, run `make html`. The files will be output in the `_build/html` directory.

Sometimes, it might be necessary to do a complete re-build of the project. To do so, run `make clean`, and then `make html`.

### 15.1.4 Submitting Changes

First, you will need to commit your changes. To do so, add your changes using `git add .`. Then, commit using:

```
git commit -m "A message describing your changes"
```

Finally, push your commit(s) to your fork using:

```
# "my-branch-name" should correspond to the name of your local branch
git push -u origin my-branch-name
```

Once your changes have been pushed, you can create a Pull Request by viewing your branch on GitHub and clicking on "Pull Request".

# CHAPTER 16

# Design Documents

This section contains documents about the design of internal features (current or future) in Glowstone.

## 16.1 1.13 Update

**Contents**



### 16.1.1 Block Format

"Block Data" refers to information about a certain block. The two components of Block Data are its **type** (e.g. air, grass, acacia door, etc.) and its **state**. Each type has a set of **properties**. Each combination of all the type's properties is a state, and has its own numerical ID.

Example:

```
"minecraft:light_gray_bed": {
    "properties": {
      "facing": [   // Possible values for the "facing" property
        "north",
        "south",
        "west",
        "east"
      ],
```

(continues on next page)

(continued from previous page)

```
      "occupied": [ // Possible values for the "occupied" property
        "true",
        "false"
      ],
      "part": [     // Possible values for the "part" property
        "head",
        "foot"
      ]
    },
    "states": [
      {
        "properties": {
          "facing": "north",
          "occupied": "true",
          "part": "head"
        },
        "id": 876
      },
      {
        "properties": {
          "facing": "north",
          "occupied": "true",
          "part": "foot"
        },
        "id": 877
      },
      [...]
    ]
  }
}
```

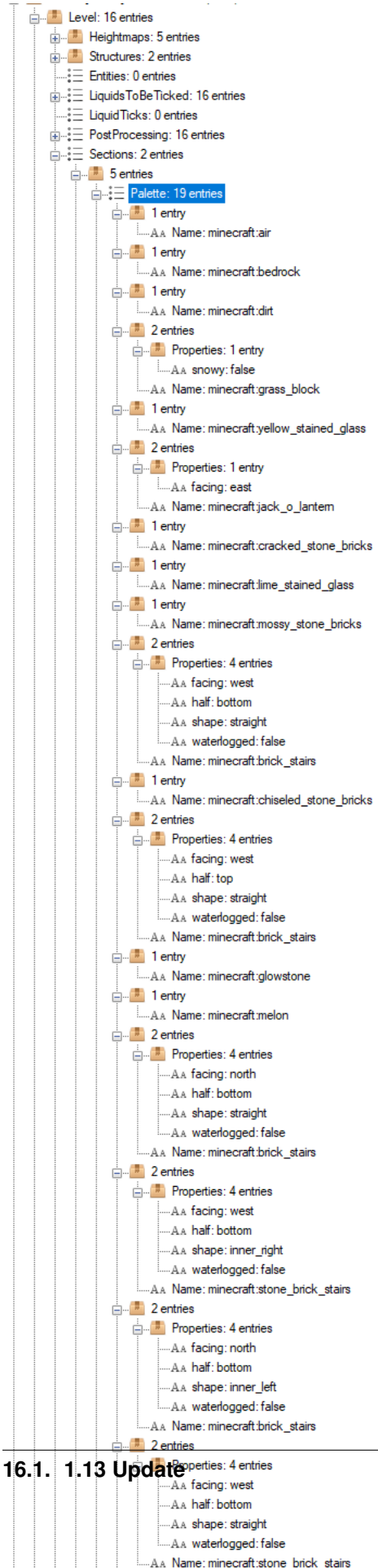
There are 2 ways to encode Block Data:

- Numerical IDs (type base ID + state bit mask)
- Map (type's `minecraft:...` key + map of string:string for state)

### On-Disk Block Storage

Each chunk section (16x16x16 blocks) has its own "Palette". A Palette is a list of Block Data maps (see above). In NBT format, the `Palette` tag is a list of compounds.

This is an example palette for a section with 19 different block states:


Level: 16 entries
Heightmaps: 5 entries
Structures: 2 entries
Entities: 0 entries
LiquidsToBeTicked: 16 entries
LiquidTicks: 0 entries
PostProcessing: 16 entries
Sections: 2 entries
5 entries
Palette: 19 entries
1 entry
Name: minecraft:air
1 entry
Name: minecraft:bedrock
1 entry
Name: minecraft:dirt
2 entries
Properties: 1 entry
snowy: false
Name: minecraft:grass_block
1 entry
Name: minecraft:yellow_stained_glass
2 entries
Properties: 1 entry
facing: east
Name: minecraft:jack_o_lantern
1 entry
Name: minecraft:cracked_stone_bricks
1 entry
Name: minecraft:lime_stained_glass
1 entry
Name: minecraft:mossy_stone_bricks
2 entries
Properties: 4 entries
facing: west
half: bottom
shape: straight
waterlogged: false
Name: minecraft:brick_stairs
1 entry
Name: minecraft:chiseled_stone_bricks
2 entries
Properties: 4 entries
facing: west
half: top
shape: straight
waterlogged: false
Name: minecraft:brick_stairs
1 entry
Name: minecraft:glowstone
1 entry
Name: minecraft:melon
2 entries
Properties: 4 entries
facing: north
half: bottom
shape: straight
waterlogged: false
Name: minecraft:brick_stairs
2 entries
Properties: 4 entries
facing: west
half: bottom
shape: inner_right
waterlogged: false
Name: minecraft:stone_brick_stairs
2 entries
Properties: 4 entries
facing: north
half: bottom
shape: inner_left
waterlogged: false
Name: minecraft:brick_stairs
2 entries
Properties: 4 entries
facing: west
half: bottom
shape: straight
waterlogged: false
Name: minecraft:stone_brick_stairs

The `BlockStates` tag is a Long Array. Each block is represented by a Palette index, corresponding to the order of the section's Palette tag. All indices have the same bit size, corresponding to the size required for the largest index (`ceil(log(n)/log(2))`, where `n` is the number of states in the Palette), with a minimum of 4 bits per index.

The number of longs in the tag may grow from 256 longs (16 indices per long) to as many longs as necessary to store all 4096 identically-sized indices.

### Network Representation

Similarly to the Disk Storage, a palette can be used to save bandwidth. The palette is a list of integers (VarInt) – the numerical IDs of each state used in the chunk.

Numerical IDs are incremental and start at 0 (`minecraft:air`). Contrary to previous Minecraft versions, these IDs are not incremental per-type, but rather per-state.

For example, if type A has 2 properties with 2 possible values each, type A has 4 possible states. If the first A state has ID 10, type B's states will start at 14.

**Encoding Example**:

```
"minecraft:A": {
  "properties": {
    "foo": [
      "true",
      "false"
    ],
    "bar": [
      "true",
      "false",
      "maybe"
    ]
  },
  "states": [
    {
      "properties": {
        "foo": "true",
        "bar": "true"
      },
      "id": 10
    },
    {
      "properties": {
        "foo": "true",
        "bar": "false"
      },
      "id": 11
    },
    {
      "properties": {
        "foo": "true",
        "bar": "maybe"
      },
      "id": 12
    },
    {
      "properties": {
        "foo": "false",
        "bar": "true"
```

(continues on next page)

(continued from previous page)

```
      },
      "id": 13
    },
    [...]
  ]
}
```

Knowing the ID of the first state of a type, it is possible to encode a state from the values of the properties.

Let's say we want to encode `minecraft:A[foo=true,bar=maybe]` to the state's ID (12):

1. From a pre-calculated map, we can determine the base ID corresponding to `minecraft:A` to be **10**.
2. The state is encoded using a recursive algorithm:
   i. A temporary value, *P*, is set to 0. This will be the final result once the algorithm is complete.

   ```
   P := 0
   ```

   ii. The `foo` property has 2 possible values ($len_{foo} = 2$). The index of the value ("true") is $i = 0$. *P* is set to the result of the following formula:

   ```
   P = i + len * P
   ```

   iii. Repeat the previous step for each property. The result for the given example will be $P = 2$:

   ```
   P = 2 + (0 + 2 * 0) * 3 = 2
   ```

   iv. The result is then *base ID* + *P* = **10 + 2 = 12**.

   A sample implementation of this algorithm can be found here (JavaScript): https://gist.github.com/momothereal/27442d5cf3b5d9ceee679b606facdfe4

**Note:** Decoding numerical IDs is not necessary on the server-side in 1.13.