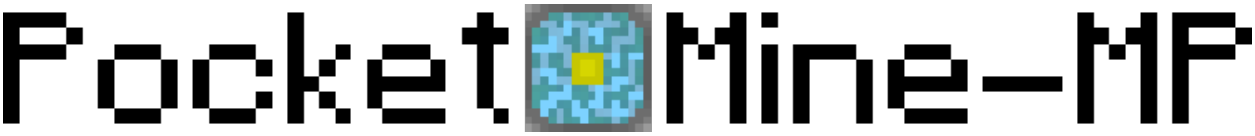
PocketMine-MP Documentation

Release 4.0.0

PocketMine Team

Sep 16, 2022

GETTING STARTED



Plugin Repository • Forums • Discord • Source Code

PocketMine-MP is a custom server software for the Minecraft: Bedrock family of Minecraft editions (includes Android, iOS, W10 and others).

- A powerful plugin API, which allows you to extend and customize your server far more easily and extensively than any competing server implementations, including the official vanilla server.
- Multi-world support, allowing you to offer a more varied game experience to players without transferring them to other server nodes.
- Performance fit to hold 100+ players (depends on hardware, see the *Setup requirements* section).
- Continuously updated to support latest Minecraft versions. PocketMine-MP has the longest and best track record of any custom server for compatibility with new Minecraft versions.

Note: PocketMine-MP is **NOT** a complete vanilla server, and it doesn't have some features you would find in the vanilla game.

If you just want to play *survival multiplayer* and don't care about *plugins*, you should consider using the official Minecraft: Bedrock server software instead of using PocketMine-MP.

CHAPTER

ONE

INSTALLATION & UPDATING

1.1 Installation

1.1.1 Setup requirements

A computer or device with the following is required:

- 64-bit CPU
- 64-bit operating system
- 1GB RAM or better

The following are not requirements, but recommended:

- Dual-core or better CPU

Note: PocketMine-MP is notoriously bad at multi-core usage. If purchasing a machine to run PocketMine-MP on, prefer higher CPU frequency instead of lots of cores.

We officially try to support Windows, Linux and MacOS platforms. However, in general any platform which will run 64-bit PHP with the required extensions will work.

1.1.2 Using https://get.pmmp.io (Linux/MacOS only)

Warning: Only works on Linux or MacOS.

Create a directory which you want to install PocketMine-MP into, and `cd` into it.

Then use `curl` to install PocketMine-MP using the following command:

```
curl -sL https://get.pmmp.io | bash -s -
```

or, if you don't have `curl`, try `wget`:

```
wget -q -O - https://get.pmmp.io | bash -s -
```

```
[*] Found PocketMine-MP Final_1.5dev (build 1254) using API 1.12.0
[*] This development build was released on Sat Jun 20 09:45:04 CEST 2015
[*] Installing/updating PocketMine-MP on directory ./
[1/3] Cleaning...
[2/3] Downloading PocketMine-MP Final_1.5dev-1254 phar... done!
[3/3] Obtaining PHP: detecting if build is available...
[3/3] MacOS 64-bit PHP build available, downloading PHP_5.6.10_x86-64_MacOS.tar.gz...␣
↪checking... regenerating php.ini... done
[*] Everything done! Run ./start.sh to start PocketMine-MP
```

Error: It is recommended to run it as a **normal user** as it doesn't need further permissions.

Do not run the installer as root, this is discouraged.

Note: If the installer doesn't work for you, try *installing manually*.

1.1.3 Installing manually

No installer available for your platform? Did the installer fail? It is not your taste? YOLO? DIY!

Getting PHP for your server

1. Download your flavor PHP binary (*Downloads*)
2. Extract the PHP binary into your server directory. If everything went well, you should have a `bin` folder in your server directory.
3. (Windows only) Download and install Microsoft Visual C++ Redistributable 2019 (*Downloads*)

Getting PocketMine-MP

1. Create a new directory for PocketMine-MP.
2. Download PocketMine-MP.phar (*Downloads*)
3. Rename the .phar to `PocketMine-MP.phar`.
4. Place it in the PocketMine-MP directory you just created.
5. Get the start script for your platform (Windows, Linux/MacOS bash)
6. (Linux/MacOS only) Make start.sh executable (chmod +x start.sh)

1.2 Updating

1.2.1 Update using https://get.pmmp.io (Linux/MacOS only)

You can use get.pmmp.io to update as well as install from scratch.

See the section *Using https://get.pmmp.io (Linux/MacOS only)*.

1.2.2 Update manually

Update PHP binary

1. Download the PHP binary for your OS (*Downloads*)
2. Delete the `bin` directory in your server folder.
3. Extract the new PHP binary. You should see a new `bin` directory has been created.

Updating PocketMine-MP

1. Delete your current PocketMine-MP.phar
2. Download the updated PocketMine-MP phar you want to use (*Downloads*)
3. Change the name to `PocketMine-MP.phar`
4. Place it in the server folder

Note: Don't forget to rename the file to `PocketMine-MP.phar`

1.3 Downloads

1.3.1 Downloads for manual install/update

PocketMine-MP prebuilt phars

- Latest stable release
- List of all releases

Tip: Can't see the downloads? Click on 'Assets' to expand the downloads list.

Prebuilt PHP binaries and related packages

Linux, Windows & MacOS

- PHP 8.0
- (Windows only) Microsoft Visual C++ 2019 Redistributable

Note: If there are no prebuilt binaries of the version you want available for your platform, you may be able to build your own using our compile scripts.

CHAPTER

TWO

BASIC USAGE

2.1 Starting the server

- Linux/MacOS: run `./start.sh`
- Windows: Double-click `start.cmd`, or open PowerShell in the server directory and run `.\start.ps1`.

2.2 Setup Wizard

The first time PocketMine-MP starts, it launches a setup wizard.

The setup wizard will allow you to choose a language, and guide you through setting up basic information about your new server, like its name, the server port, etc.

Note: You'll be asked to accept the terms of PocketMine-MP's license. You can read the full text of the license on GitHub.

Tip: You can skip the setup wizard by passing `--no-wizard` to `start.sh`, `start.cmd` or `start.ps1`.

2.3 Stopping the server

To stop the server, simply type `stop` in the console and press enter.

Error: Do NOT click the X to stop the server. You could lose data or your data might get corrupted.

CHAPTER

THREE

CONFIGURATION

3.1 Server behaviour

PocketMine-MP's behaviour is controlled by several configuration files. You can edit them to change the behaviour of your server.

- `server.properties` contains basic settings like the server name, port, maximum view distance, etc. These settings are all safe to change.
- `pocketmine.yml` contains more advanced settings settings like memory usage, max thread count, etc. It also contains settings for loading multiple worlds.

Warning: It's best to **leave a setting alone** if you don't understand what it's for. Many settings in `pocketmine.yml` can break your server if configured incorrectly.

Note: To edit configs on Windows, right-click on the file → Open With → Choose another app → Notepad.

3.2 Resource packs

`resource_packs.yml` is found in the `resource_packs` folder.

Read more about installing resource packs.

3.3 Player permissions

There are several files that allow you to control player permissions on your server:

- `ops.txt` is a simple list of player names that have "op" permissions on your server. Ops can do more things than regular players, like giving items, teleporting, stopping the server and more.
- `banned-players.txt` lists names of players that are banned from your server.
- `banned-ips.txt` lists IPs that are not allowed to connect to your server. This is useful for troublesome players who change accounts and keep coming back to hassle you.

Note: You can modify these by using slash commands instead of editing the files directly. See `/op`, `/deop`, `/ban`, `/unban`, `/ban-ip` and `/unban-ip`.

CHAPTER

FOUR

PLUGINS

PocketMine-MP can be extended and customized with plugins.

A plugin is an external module which can be "plugged in" to your server to add custom features, change default behaviours and more.

4.1 Downloading plugins

You can find a wide range of premade plugins on the Poggit Release plugin platform.

4.2 Installing/updating a plugin

Plugin developers usually distribute plugins as phar files. These files can be easily loaded by placing them in your `plugins` folder in the server data folder.

Be sure to **delete old versions of the plugin**, and **restart the server** to see changes.

4.3 Writing your own plugins

You can find resources for plugin development at the DeveloperDocs site.

CHAPTER

FIVE

RESOURCE PACKS & BEHAVIOUR PACKS

PocketMine-MP has some support for resource packs.

Any number of valid resource packs may be loaded. However, per-world resource packs are not currently supported at the time of writing.

Error: Behaviour packs are **not** supported.

Warning: Resource packs which **depend on behaviour packs** may not work correctly. While PocketMine-MP will not prevent you loading such packs, it makes no attempt to validate dependencies and your game may experience errors or crashes when attempting to use them.

5.1 Installing resource packs

Resource packs to be loaded should be placed in the `resource_packs` folder of your server data.

Then, you will need to add them to the stack in the `resource_packs.yml` file (also located in the `resource_packs` folder).

5.2 Resource stack

This is a list of resource packs in `resource_packs.yml` that will used by your server. It should contain a list of valid resource pack file names found in the `resource_packs` folder, including the file extension.

Resource packs are applied from bottom to top, similar to how they work in Minecraft itself. This means that resources at the top of the list will override resources lower down the list.

5.3 Supported pack formats

At the time of writing, the following types of resource pack are supported:

- `.zip`
- `.mcpack`

Error: Unpacked (folder) resource packs are not currently supported.

CHAPTER

SIX

PERMISSIONS

PocketMine-MP includes a powerful permission system, similar in functionality to the one found in Bukkit. It allows fine-grained control of access to various parts of the server functionality, such as:

- Ability to use individual commands
- Whether or not the user can see administrative broadcasts when a command is used (e.g. when someone uses `/op`)

In addition, plugins can offer permissions to allow customising access to behaviours that they offer.

PocketMine-MP doesn't directly offer any way to use permissions at the time of writing (August 2022), but you can find permission management plugins in the "Admin Tools" section on Poggit.

6.1 Built-in permissions

6.1.1 PocketMine-MP Core Permissions

Generated from PocketMine-MP 4.7.1+dev

Name	Description	Implied permi
`pocketmine.broadcast.admin`	Allows the user to receive administrative broadcasts	N/A
`pocketmine.broadcast.user`	Allows the user to receive user broadcasts	N/A
`pocketmine.command.ban.ip`	Allows the user to ban IP addresses	N/A
`pocketmine.command.ban.list`	Allows the user to list banned players	N/A
`pocketmine.command.ban.player`	Allows the user to ban players	N/A
`pocketmine.command.clear.other`	Allows the user to clear inventory of other players	N/A
`pocketmine.command.clear.self`	Allows the user to clear their own inventory	N/A
`pocketmine.command.defaultgamemode`	Allows the user to change the default gamemode	N/A
`pocketmine.command.difficulty`	Allows the user to change the game difficulty	N/A
`pocketmine.command.dumpmemory`	Allows the user to dump memory contents	N/A
`pocketmine.command.effect`	Allows the user to give/take potion effects	N/A
`pocketmine.command.enchant`	Allows the user to enchant items	N/A
`pocketmine.command.gamemode`	Allows the user to change the gamemode of players	N/A
`pocketmine.command.gc`	Allows the user to fire garbage collection tasks	N/A
`pocketmine.command.give`	Allows the user to give items to players	N/A
`pocketmine.command.help`	Allows the user to view the help menu	N/A
`pocketmine.command.kick`	Allows the user to kick players	N/A
`pocketmine.command.kill.other`	Allows the user to kill other players	N/A
`pocketmine.command.kill.self`	Allows the user to commit suicide	N/A

continues on nex

Table 1 – continued from previous page

Name	Description	Implied permi
pocketmine.command.list	Allows the user to list all online players	N/A
pocketmine.command.me	Allows the user to perform a chat action	N/A
pocketmine.command.op.give	Allows the user to give a player operator status	N/A
pocketmine.command.op.take	Allows the user to take a player's operator status	N/A
pocketmine.command.particle	Allows the user to create particle effects	N/A
pocketmine.command.plugins	Allows the user to view the list of plugins	N/A
pocketmine.command.save.disable	Allows the user to disable automatic saving	N/A
pocketmine.command.save.enable	Allows the user to enable automatic saving	N/A
pocketmine.command.save.perform	Allows the user to perform a manual save	N/A
pocketmine.command.say	Allows the user to talk as the console	N/A
pocketmine.command.seed	Allows the user to view the seed of the world	N/A
pocketmine.command.setworldspawn	Allows the user to change the world spawn	N/A
pocketmine.command.spawnpoint	Allows the user to change player's spawnpoint	N/A
pocketmine.command.status	Allows the user to view the server performance	N/A
pocketmine.command.stop	Allows the user to stop the server	N/A
pocketmine.command.teleport	Allows the user to teleport players	N/A
pocketmine.command.tell	Allows the user to privately message another player	N/A
pocketmine.command.time.add	Allows the user to fast-forward time	N/A
pocketmine.command.time.query	Allows the user query the time	N/A
pocketmine.command.time.set	Allows the user to change the time	N/A
pocketmine.command.time.start	Allows the user to restart the time	N/A
pocketmine.command.time.stop	Allows the user to stop the time	N/A
pocketmine.command.timings	Allows the user to record timings to analyse server performance	N/A
pocketmine.command.title	Allows the user to send a title to the specified player	N/A
pocketmine.command.transferserver	Allows the user to transfer self to another server	N/A
pocketmine.command.unban.ip	Allows the user to unban IP addresses	N/A
pocketmine.command.unban.player	Allows the user to unban players	N/A
pocketmine.command.version	Allows the user to view the version of the server	N/A
pocketmine.command.whitelist.add	Allows the user to add a player to the server whitelist	N/A
pocketmine.command.whitelist.disable	Allows the user to disable the server whitelist	N/A
pocketmine.command.whitelist.enable	Allows the user to enable the server whitelist	N/A
pocketmine.command.whitelist.list	Allows the user to list all players on the server whitelist	N/A
pocketmine.command.whitelist.reload	Allows the user to reload the server whitelist	N/A
pocketmine.command.whitelist.remove	Allows the user to remove a player from the server whitelist	N/A
pocketmine.group.console	Grants all console permissions	*Jump*
pocketmine.group.operator	Grants all operator permissions	*Jump*
pocketmine.group.user	Grants all non-sensitive permissions that everyone gets by default	*Jump*

Implied permissions

Some permissions automatically grant (or deny) other permissions by default when granted. These are referred to as **implied permissions**.

Permissions may imply permissions which in turn imply other permissions (e.g. `pocketmine.group.operator` implies `pocketmine.group.user`, which in turn implies `pocketmine.command.help`).

Implied permissions can be overridden by explicit permissions from elsewhere.

Note: When explicitly denied, implied permissions are inverted. This means that "granted" becomes "denied" and vice versa.

Permissions implied by `pocketmine.group.console`

Users granted this permission will also be granted/denied the following permissions implicitly:

Name	Type
`pocketmine.command.dumpmemory`	Granted
`pocketmine.group.operator`	Granted

Permissions implied by `pocketmine.group.operator`

Users granted this permission will also be granted/denied the following permissions implicitly:

Name	Type
`pocketmine.broadcast.admin`	Granted
`pocketmine.command.ban.ip`	Granted
`pocketmine.command.ban.list`	Granted
`pocketmine.command.ban.player`	Granted
`pocketmine.command.clear.other`	Granted
`pocketmine.command.defaultgamemode`	Granted
`pocketmine.command.difficulty`	Granted
`pocketmine.command.effect`	Granted
`pocketmine.command.enchant`	Granted
`pocketmine.command.gamemode`	Granted
`pocketmine.command.gc`	Granted
`pocketmine.command.give`	Granted
`pocketmine.command.kick`	Granted
`pocketmine.command.kill.other`	Granted
`pocketmine.command.list`	Granted
`pocketmine.command.op.give`	Granted
`pocketmine.command.op.take`	Granted
`pocketmine.command.particle`	Granted
`pocketmine.command.plugins`	Granted
`pocketmine.command.save.disable`	Granted
`pocketmine.command.save.enable`	Granted
`pocketmine.command.save.perform`	Granted
`pocketmine.command.say`	Granted
`pocketmine.command.seed`	Granted
`pocketmine.command.setworldspawn`	Granted
`pocketmine.command.spawnpoint`	Granted
`pocketmine.command.status`	Granted
`pocketmine.command.stop`	Granted
`pocketmine.command.teleport`	Granted
`pocketmine.command.time.add`	Granted
`pocketmine.command.time.query`	Granted
`pocketmine.command.time.set`	Granted
`pocketmine.command.time.start`	Granted
`pocketmine.command.time.stop`	Granted
`pocketmine.command.timings`	Granted
`pocketmine.command.title`	Granted

continues on next page

Table 2 – continued from previous page

Name	Type
pocketmine.command.transferserver	Granted
pocketmine.command.unban.ip	Granted
pocketmine.command.unban.player	Granted
pocketmine.command.whitelist.add	Granted
pocketmine.command.whitelist.disable	Granted
pocketmine.command.whitelist.enable	Granted
pocketmine.command.whitelist.list	Granted
pocketmine.command.whitelist.reload	Granted
pocketmine.command.whitelist.remove	Granted
pocketmine.group.user	Granted

Permissions implied by pocketmine.group.user

Users granted this permission will also be granted/denied the following permissions implicitly:

Name	Type
pocketmine.broadcast.user	Granted
pocketmine.command.clear.self	Granted
pocketmine.command.help	Granted
pocketmine.command.kill.self	Granted
pocketmine.command.me	Granted
pocketmine.command.tell	Granted
pocketmine.command.version	Granted

6.2 Names

Permissions are referred to by names, which usually look something like `pocketmine.command.help`.

There's no special requirements on permission names, but the de facto standard in community plugins is to separate the parts using `.`, and use a common prefix for permissions for similar things. For example all PocketMine-MP core commands have permissions using the prefix `pocketmine.command.`.

6.3 Grouping permission nodes together

While most permissions affect one specific thing (e.g. whether or not a player is allowed to use a command), permissions may also automatically grant (or deny) other permissions when assigned. These special permissions are known as "group permissions".

An example of a group permission is `pocketmine.group.operator`, which, when granted, grants permissions such as `pocketmine.command.op.give` which enable access to operator commands.

This is useful if you want to assign the same set of permissions to multiple users (or other groups).

6.4 Permission precedence

Because of group permissions, it's possible that a user may receive multiple conflicting values for the same permission.

Permission values are resolved in the following order:

- Explicitly overridden permissions take maximum priority. This means that, for example, a player can receive `pocketmine.group.operator`, but be denied `pocketmine.command.op.give`, allowing them access to all operator commands **except** `/op`.
- If the permission is not overridden directly, the value from the most recently assigned group permission will be used.

CHAPTER

SEVEN

CONTACT AND SUPPORT

7.1 Help & Support

Can't find what you're looking for here? Whether it's help setting up a server, finding problems with plugins, or anything else related to PocketMine-MP, our community will be happy to help you.

- Discord
- Forums

7.2 Reporting Bugs

- Contact team@pmmp.io directly if you want to report a security issue / exploit.
- For all other bugs, head to our GitHub issue tracker.

CHAPTER

EIGHT

USEFUL LINKS

- PMMP GitHub organisation
- PocketMine-MP GitHub repository
- PHP compile scripts
- Linux & MacOS install script source
- PMMP Website
- PMMP Forums
- PocketMine-MP Translation Project

CHAPTER

NINE

INSTALLATION

9.1 Failed loading opcache.so (or other PHP extensions)

This may happen when the installer is not used or when PocketMine-MP was moved.

To fix this issue, run the following from wherever your PHP bin directory is:

```
EXTENSION_DIR=$(find "$(pwd)/bin" -name *debug-zts*)
grep -q '^extension_dir' bin/php7/bin/php.ini && sed -i'bak' "s{^extension_dir=.*
↪{extension_dir=\"$EXTENSION_DIR\"{" bin/php7/bin/php.ini || echo "extension_dir=\"
↪$EXTENSION_DIR\"" >> bin/php7/bin/php.ini
```

This will locate your PHP binary's extension_dir on the disk and set it into php.ini, replacing it if it already exists, and adding it if not.

9.2 PocketMine-MP.phar not found when running server

Troubleshooting tips:

- Make sure you downloaded a PocketMine-MP phar file and put it in your server folder.
- Check that the name is exactly `PocketMine-MP.phar` - no spaces or any other artifacts.
- Check that the name casing is correct. `PocketMine-MP.phar` will work, but `pocketmine-mp.phar` won't.
- Check that the file extension is not duplicated. `PocketMine-MP.phar.phar` won't work.

9.3 Can't install as user root

> **Warning:** Running the installer as root is **strongly discouraged**.
>
> Bugs in the installer have previously caused **loss of data** for people who ran it as root.
>
> It is recommended to run it as a normal user as it doesn't need further permissions.

We recommend you to install PocketMine-MP as a normal user, not as root. Create one if you don't have one.

```
useradd -d /home/pocketmine -m pocketmine
passwd pocketmine
```

9.4 Can I install PocketMine-MP on Windows XP?

PocketMine-MP can not be installed on Windows XP. Is it an old computer? Try Linux!

CHAPTER

TEN

CONNECTING

10.1 Can't connect to the server after updating Minecraft

Newer versions of Minecraft Bedrock often bring breaking network changes unpredictably. If you are unable to connect after updating your version, it is likely that you need to update your server version to a newer one that supports the version you're trying to connect with.

Often, **but not always**, newer patch versions (e.g. 1.2.7, 1.2.8, 1.2.9) are compatible without any update needed. Look for the latest version which offers a version less than or equal to yours.

Run the `version` command to check the supported Minecraft version of your server.

Example:

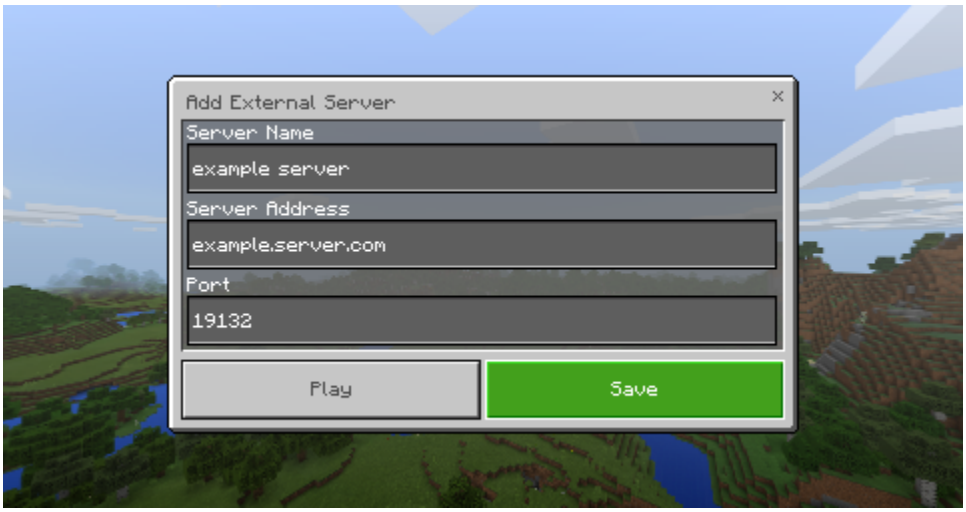
```
for Minecraft: PE v1.2.7 (protocol version 160)
```

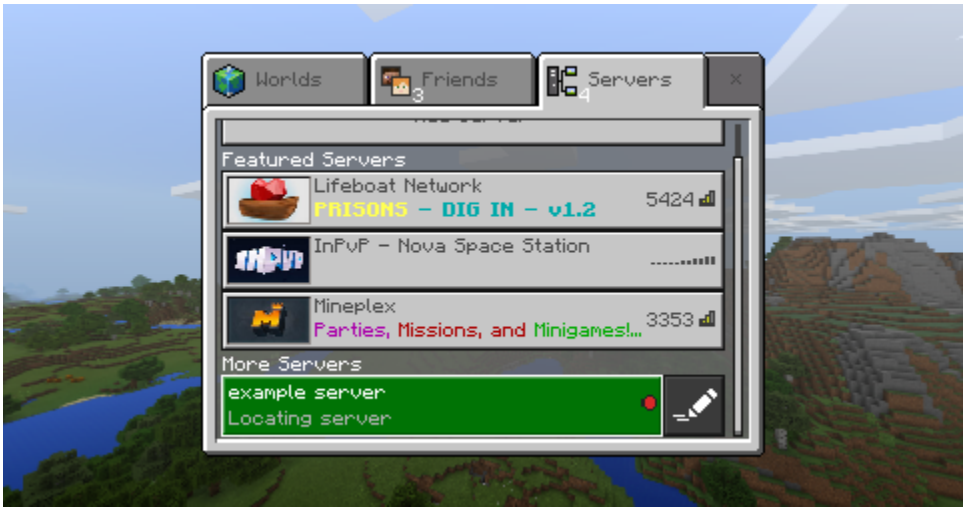
10.2 What does "Opening server on 0.0.0.0:19132" mean?

`0.0.0.0` means "all IPv4 addresses on the local machine". If a host has two ip addresses, `192.168.1.1` and `10.1.2.1`, and a server running on the host listens on `0.0.0.0`, it will be reachable at **both** of those IPs.

10.3 How do I add an external server which is not on my network?

To connect to a server by IP/address and port, you need to add it to the Servers list.





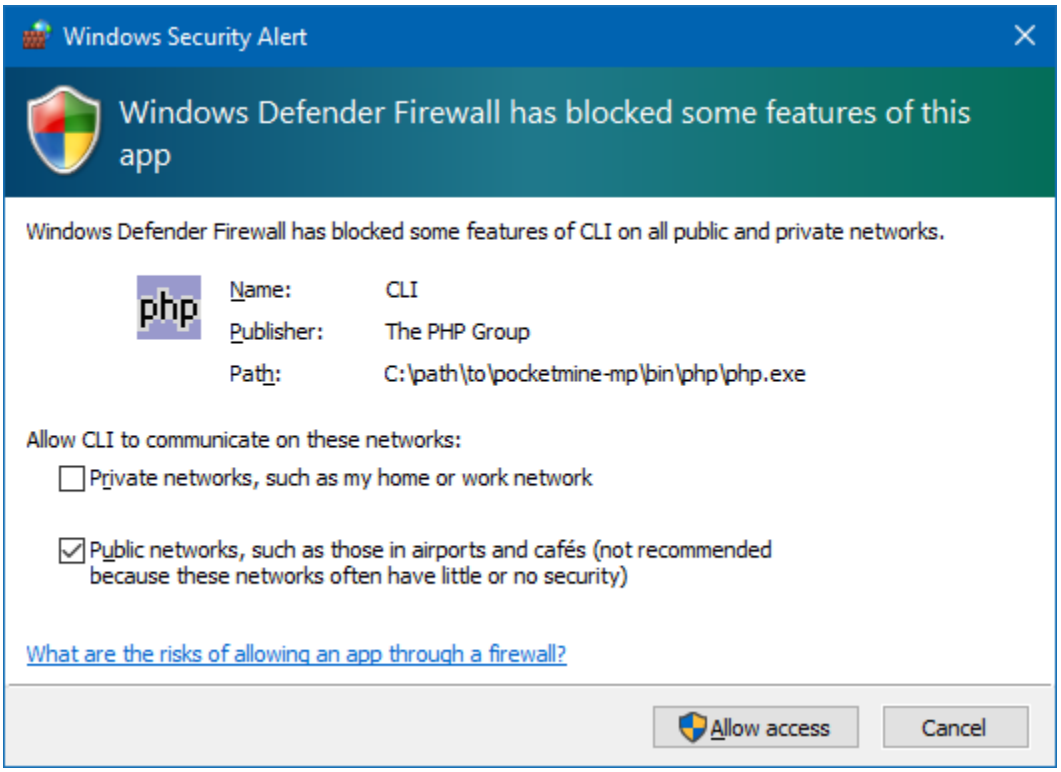
Note: A local server should show up on the `Friends` tab without adding the details.

10.4 Do I have to open ports in my firewall?

If you have a firewall set up then you need to allow access to `UDP port 19132`.

Note: Do you want to use **RCON**? If so, then `TCP port 19132` also needs to be open.

Note: On Windows, you might get a dialog like this when you first start the server. Click "Allow access" to allow PocketMine-MP through the firewall and allow players to connect.



10.5 Can other users connect to my server?

Users on the same network are able to join the server. If you want other people from outside your own network to be able to join then you need to port-forward.

10.6 Do I have to configure port forwarding?

This is only needed when you want people from outside your network to connect. Check portforward.com or use Google to find the instructions. Use the brand and type of your router as keywords.

Note:

- UDP port: 19132 for PocketMine-MP and Query
- TCP port: 19132 for RCON

10.7 RakLib Thread Blocked x.x.x.x for x seconds

Timeout	Reason
5 seconds	Error on a normal MCPE packet
300 seconds	More than 5000 packets per tick from one address
600 seconds	Errors on external packet (like Query or RCON)

10.8 VPN connection and gameplay issues

Minecraft Bedrock commonly has trouble with VPNs for reasons which are not entirely clear.

10.8.1 Make sure all MTU limits are set correctly

A common cause of VPN issues is incorrect, inconsistent or too-large tunnel or link MTUs. You can try testing with smaller MTU sizes and see if the problem goes away.

10.8.2 Reduce maximum MTU in PocketMine-MP itself

If you are not able to alter your VPN MTU sizes, you can try reducing the `network.max-mtu-size` setting in your `pocketmine.yml` file. The default setting is 1492 - you may have success with smaller sizes.

Note: While a smaller MTU may fix your connection issues, it also reduces efficiency, which may cause higher bandwidth consumption and higher data usage.

10.9 Minecraft can't connect to a server on the same computer on Windows

Note: These steps are **not** required to connect to all servers. You only need to do this if you're connecting to a server on the **same computer** that you're running Minecraft on.

10.9.1 Disable UWP loopback restrictions for Minecraft

Windows UWP apps have some restrictions on what they are allowed to do by default. Unfortunately, this includes connecting to things on localhost.

To lift this restriction from Minecraft, launch Windows PowerShell as an administrator and run the following:

```
CheckNetIsolation LoopbackExempt -a -n="Microsoft.MinecraftUWP_8wekyb3d8bbwe"
```

For Minecraft Preview, you'll need a slightly different command:

```
CheckNetIsolation LoopbackExempt -a -n="Microsoft.MinecraftWindowsBeta_8wekyb3d8bbwe"
```

If everything goes well, you'll see a message: `OK.` You should now be able to connect (you may need to close and reopen the game).

10.9.2 Don't use `localhost` for server address

Minecraft for Windows doesn't correctly handle the `localhost` address.

Try using `127.0.0.1` (IPv4) or `::1` (IPv6) instead when adding the server to your server list.

CHAPTER

ELEVEN

PLAYING

11.1 Why doesn't X or Y gameplay feature work on PocketMine-MP? Is it a bug?

PocketMine-MP does not have all gameplay features that Minecraft itself offers. This is because PocketMine-MP is developed by developers in their spare time who have difficulty keeping up with new Minecraft features.

Most notably, current missing features at the time of writing include mobs, redstone, minecarts and dimensions. These will be added in the future.

11.2 Can Minecraft: Java Edition (PC) clients connect to a PocketMine-MP server?

No, but plugins exist which add partial support for this. Look up "BigBrother" on GitHub.

11.3 Unable to build without OP permissions

This is usually caused by the built-in spawn protection. By default it protects a circle of radius 16 blocks around the spawn point. Try moving more than 16 blocks from the spawn and see if you're able to build.

You can disable or configure the spawn protection using the `spawn-protection` setting in your `server.properties`. Set it to `-1` to turn it off.

CHAPTER

TWELVE

PLUGINS

12.1 What does "Incompatible API version" mean when loading a plugin?

This means that the version of the plugin currently installed is not compatible with the server. Check for an updated version of the plugin, ask the plugin developer to update it, or try updating it yourself.

PocketMine-MP's API frequently undergoes breaking and substantial changes. The API version exists as a way for a plugin and server to determine whether the plugin can work correctly on the server's current API.

Warning: Simply bumping the plugin's declared API version is often not enough to update a plugin. The plugin might still crash or not work as expected. Use of so-called "API updaters" is discouraged.

12.2 I can't get a `.phar` for a plugin. How do I create one?

You can create `.phar` files from plugin source code using the DevTools plugin. Instructions on how to build a phar from source code are given on the README.md.

12.3 Can I run a plugin from source without creating a `.phar`?

You can use the DevTools plugin to load source plugins (known as "folder plugins").

Warning: It is discouraged to use either DevTools or folder plugins on a production server.

For small test plugins there is a new way, check out this forum thread

CHAPTER

THIRTEEN

ABOUT POCKETMINE-MP

13.1 What's the difference between PMMP and PocketMine?

In late 2015, *Shoghi Cervantes* was forced to stop developing PocketMine-MP due to conflicts with his job at Mojang. He had exclusive administrator privileges over almost all of the legacy PocketMine infrastructure, including the old PocketMine GitHub organization, the old PocketMine Forums, the old PocketMine website, and more besides.

Since he fell out of contact with everyone involved with PocketMine, the remaining team members were unable to maintain infrastructure, properly moderate the forums, or add new GitHub collaborators to maintain the code.

In September 2016, a collaboration of several members of the old PocketMine team, plus a new developer Dylan (@dktapps), formed a new organization called PMMP. To work around the inability to work on the old infrastructure, the following things were created:

- PMMP GitHub organization
- PMMP Forums
- PMMP website

and more besides.

In late 2017, *Shoghi Cervantes* endorsed PMMP as PocketMine's successor on Twitter, and redirected many parts of the pocketmine.net domain to their corresponding parts of pmmp.io. The old PocketMine Forums remains for historical purposes.

Shoghi Cervantes also granted the PMMP team access to the PocketMine GitHub organization at that time, but the team chose not to use it since the PMMP GitHub organization had at that point been established for over a year and had already developed a lot of history of its own, such as issues, which were difficult to migrate at the time, and instead archived all the repositories and added redirections to the new PMMP GitHub organization.

13.2 Who is @shoghicp?

@shoghicp (Shoghi Cervantes) is the creator of PocketMine-MP (originally Pocket Minecraft PHP).

Shoghi developed the project from October 2012 until January 2016. He was hired by Mojang to work on Minecraft PE in 2014.

He was forced to stop developing PocketMine-MP due to conflicts with his job developing Minecraft PE at Mojang.

13.3 Why PHP?

Back in its early days, PocketMine-MP (at the time known as Pocket Minecraft PHP) was intended as a **quick prototype** for reverse engineering the Minecraft PE protocol by its creator, Shoghi Cervantes. It was **never intended** for use on production servers. (Tweet from @shoghicp)

Since PocketMine-MP was the only server software available for Minecraft PE at the time, it quickly grew in popularity and started to be used by thousands of servers. Lifeboat Survival Games was one of the first big servers in Minecraft PE, and it was built using PocketMine-MP.

Today, PocketMine-MP maintains its popularity because it's the oldest and most well known Bedrock server software out there, as well as being really easy to develop plugins for.

CHAPTER

FOURTEEN

CRASHES

PocketMine-MP may crash for a number of reasons:

- Bad / faulty plugins.
- Running out of memory due to not enough or memory leak.
- Bugs within the PocketMine-MP which are unrecoverable.

In all of the above cases, a crashdump file will be generated in the `crashdumps` folder of your server.

14.1 Reporting a crash

Crash reports are automatically submitted to our Crash Archive if your server is connected to the internet.

If successful, you will see a message like this in the server log: `[18:34:15] [Server thread/EMERGENCY]: The crash dump has been automatically submitted to the Crash Archive. You can view it on https://crash.pmmp.io/view/5555 or use the ID #5555.`

If you want to submit a crash report manually, you can do so at the submit page.

Crashdumps are searched and investigated regularly by the development team, so it is not necessary to submit an issue if your crashdump is in the Crash Archive. Nonetheless, you can create one anyway at the issues page if you want to give us extra information or draw our attention to the issue.

Note: If you want to opt-out of automatic crashdump submission, you can disable it using the `auto-report.enabled` setting in your `pocketmine.yml`.

> **Warning:** When reporting a crash in PocketMine-MP, a crashdump file **must** be provided or we will not be able to help you.

14.2 What's in a crashdump file?

Crashdumps are `.log` files which contain important information which is used to identify, reproduce, and fix bugs. They contain human-readable information about the crash (the top part) and a machine-readable part which is used by our Crash Archive.

A crashdump may contain the following information:

- Error message, file name, line number and error type
- A stack trace of the error, which helps identify where the crash happened
- A sample of code around the site of the crash, which helps to identify where the crash happened
- Information about your operating system version and hardware (such as CPU model)
- List of plugins installed on the server
- Version of PocketMine-MP which you are using
- The contents of your `pocketmine.yml` and `server.properties` file (sensitive information like passwords are redacted)

Note: You can fine-tune what information is placed in crashdumps in your `pocketmine.yml` file under the `auto-report` section.

CHAPTER

FIFTEEN

USING THE GITHUB ISSUE TRACKER

Did your server crash, or did you encounter a bug?

- Make sure you're using the latest available version of PocketMine-MP, as the bug might already have been fixed.
- Try and reproduce it **WITHOUT PLUGINS**, as plugins can frequently cause issues.
- Ask for help on our forums before creating an issue.

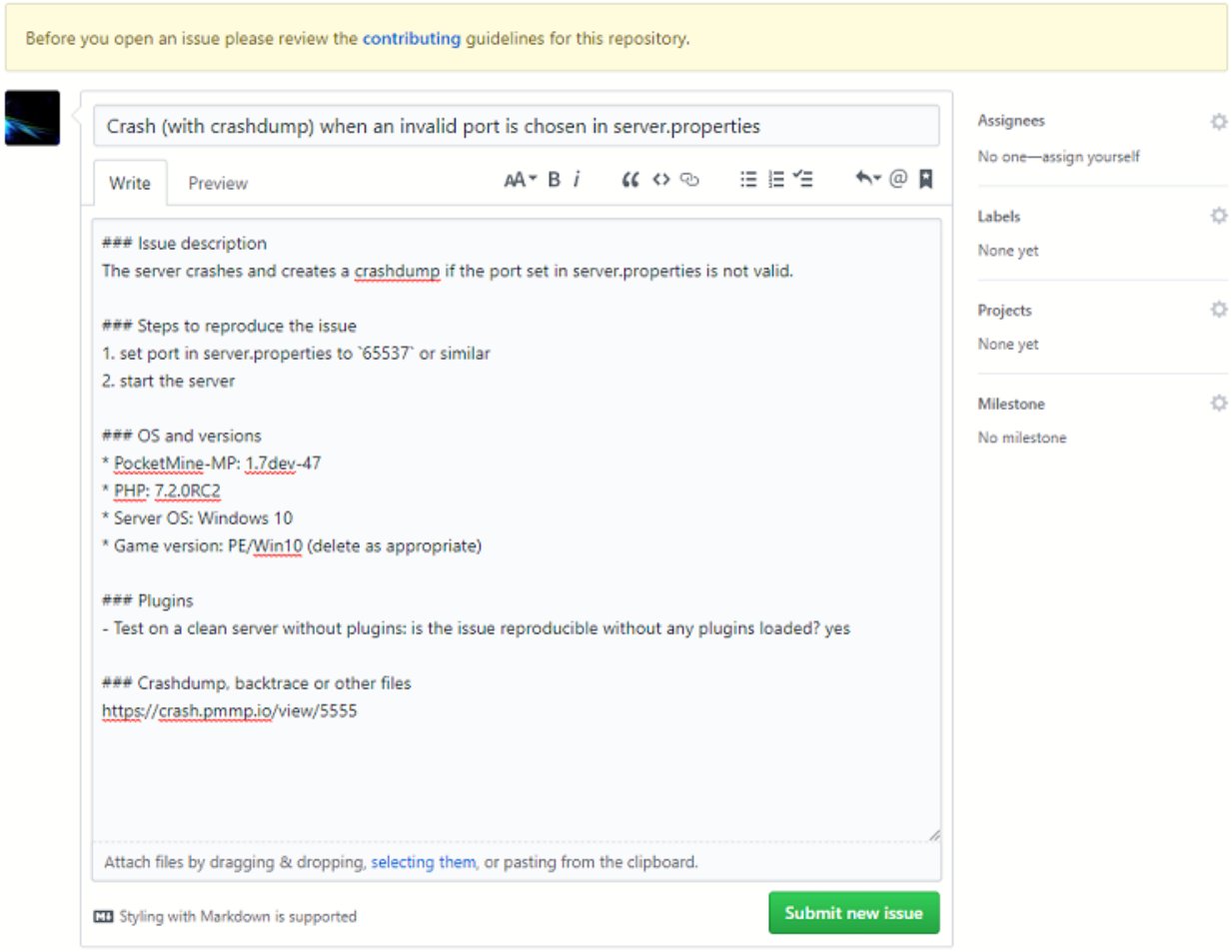
Warning: Please **do not** use our issue tracker for support requests, but instead seek assistance on the forums or our community Discord. Support request issues will be closed as per the contribution guidelines.

Note: Make sure you read the contribution guidelines before creating an issue.

If your issue is still unresolved and you're sure the issue is caused by PocketMine-MP itself, then make a new issue on GitHub.

15.1 Issue template

An issue template is provided, showing the information that we require for an issue submission. **Do not** just delete the template - fill it with the information it asks for. Give as much information as you can about when or what happened.


Before you open an issue please review the contributing guidelines for this repository.
Crash (with crashdump) when an invalid port is chosen in server.properties
Write
Preview
Issue description
The server crashes and creates a crashdump if the port set in server.properties is not valid.
Steps to reproduce the issue
1. set port in server.properties to `65537` or similar
2. start the server
OS and versions
* PocketMine-MP: 1.7dev-47
* PHP: 7.2.0RC2
* Server OS: Windows 10
* Game version: PE/Win10 (delete as appropriate)
Plugins
- Test on a clean server without plugins: is the issue reproducible without any plugins loaded? yes
Crashdump, backtrace or other files
https://crash.pmmp.io/view/5555
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Submit new issue
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone

CHAPTER

SIXTEEN

PLUGIN DEVELOPER REFERENCE

This section contains documentation and reference pages for PocketMine-MP plugin developers. This documentation is a work in progress and contributions are welcomed. See the TODO list if you need ideas what to work on.

16.1 API versioning

PocketMine-MP plugins are required to declare which API versions they are compatible. This is used to decide whether or not to load a plugin, and to gracefully degrade when the plugin is not compatible with the given server version.

As of PocketMine-MP 3.0.0, the API version is the same as the server version. This version is a semantic `major.minor.patch` version number. Read more about semantic versioning.

16.1.1 Definitions

Semver is roughly defined as the following:

- Major version bump: Breaking changes - the public API has changed in such a way that it breaks thing depending on it.
- Minor version bump: Feature additions or big changes which do not break API. This can include API methods becoming deprecated, new API features being added, but should not break plugins designed for previous minor versions.
- Patch version bump: Usually bug fixes. These shouldn't break the API nor cause any significant alteration to the description of a version.

PocketMine-MP uses a condition of `==.>=.>=` (or `eq.ge.ge` if you prefer `bash` notation) for comparing API versions. This means that:

- The major version **MUST be the same** to be compatible
- The server's minor version **MUST be AT LEAST the same** as the plugin's, although it can be greater.
- The server's patch version **MUST be AT LEAST the same** as the plugin's, but can also be greater.

The PocketMine-MP developers strive to ensure that any non-major version does not break API compatibility with plugins. This means that a plugin written to target `3.1.1` should also work on any future `3.x.y` version, but **not** `4.0.0` or any future major versions.

16.1.2 Examples

Server version	Plugin version	Compatible	Reason
4.0.0	3.0.0	NO	Major versions are different.
3.1.0	3.0.0	YES	Server has a greater minor version than the plugin, guaranteeing compatibility.
3.0.0	3.1.0	NO	Plugin requires new features not found in the given server version
3.0.1	3.0.0	YES	Server has a greater patch version than the plugin, guaranteeing compatibility.
3.0.0	3.0.1	NO	Plugin requires newer bug fixes not found in the given server version

16.1.3 How to choose your supported API version

You should choose the **minimum version that supports the things that you need**.

For example, if the feature you want was first added in `3.1.0`, you can require `3.1.0` as a minimum even if the currently supported version is `3.2.0`. This is to ensure that users get a smoother experience no matter what version of PocketMine-MP they are using.

Of course, you should test your plugin on your minimum version to make sure that everything works correctly. However, it may of course not make sense to require a lower version if the earlier versions are end-of-life.

16.1.4 Common pitfalls and misconceptions

- The API version **is not a magic number**. Changing it blindly **will not make a plugin magically work** on a version it wasn't designed for.
- It is **not required to write out every single compatible version** in your plugin's manifest. Only the **minimum required version of each major version** is required.
  - Good: `[3.1.0]` (single minimum version), `[3.2.0, 4.0.0]` (minimum version from two different major branches)
  - Bad: `[3.1.0, 3.1.1, 3.1.2]` - this is **not** desired and the superfluous versions will be ignored.
- Ensure that your plugin **actually runs on the versions you specified**. Versions which it a) crashes on, or b) doesn't work as expected, should be removed from your versions list.

16.2 Events

PocketMine-MP has an events system which allows plugins to react to, modify the outcome of, and prevent the result of events.

16.2.1 How it works

1. Something registers a handler for a given event.
2. Just before the event takes place, the handler is called and passed an object containing information about the event. This allows handlers to react to, modify (and in some cases prevent) an event from taking place.
3. The event takes place (or does not take place if cancelled) as defined by the object which contains the event information.

Note: All event handlers are currently executed **before** the event takes place. This is a common pitfall of PocketMine-MP plugin developers - when an event handler is executed, the actual event has not yet taken place.

16.2.2 Handling events

Using the `pocketmine\event\Listener` interface

This interface is implemented by classes which want to have matching methods within themselves (and their child classes) to be treated as event handlers on registration. Every method in a `Listener`-implementing class is checked as a candidate event handler.

How does it work?

When you call `registerEvents()`, PocketMine-MP does some complicated reflection magic to examine the functions declared in the given listener object. It then adds any function that meets the criteria below to the handler list for the desired event.

The type of event handled by a function is decided by the type of the first parameter.

Note: Handler function names are **completely ignored**. Therefore, it doesn't matter whether your handler is called `onPlayerJoin()` or `iEatEnglishForBreakfast()`; as long as the function meets the required criteria described below, it will be registered.

What counts as a handler method?

To be considered as an event handler candidate, a `Listener` method MUST meet the below criteria:

- MUST accept exactly ONE parameter
- This one parameter MUST be a class extending `pocketmine\event\Event` which is either non-abstract or declares the `@allowHandle` in the class doc comment.
- MUST be public
- MUST NOT be static
- MUST be declared by a class implementing `Listener` or extending a class which implements `Listener` (i.e. a method in a non-`Listener`-implementing base class which meets the above criteria IS NOT be considered a handler)
- MUST NOT declare the `@notHandler` PhpDoc annotation (will be ignored if it does)

The following are not mandatory but recommended:

- SHOULD NOT return a value (recommended to have a `void` return type (this may become a requirement in future))
- SHOULD NOT handle an event marked as `@deprecated`

Annotations controlling a handler function's behaviour

Event handlers within a `Listener` can have their behaviour altered using certain PhpDoc doc-comment tags (otherwise known as "annotations"). The following annotations are respected for candidate handlers:

- `@notHandler`: Marks a function as explicitly NOT being an event handler, even if it meets all other criteria.
- `@ignoreCancelled`: This handler WILL NOT receive events which are cancelled before reaching it.
- `@priority`: Allows controlling when in the event calling sequence this handler will be executed. This allows competing handlers of the same event to cooperate to some extent. See the section below on event priority. Example: `@priority NORMAL`
- `@softDepend`: Skips registering the handler if the event class it wants to handle does not exist. This can be used to soft-depend on classes provided by other plugins. Example: `@softDepend SimpleAuth`

Event handler priority

Event "priority" is used to control the order in which multiple handlers of the same event will execute. This is mainly used for plugin interoperability. The `pocketmine\event\EventPriority` class declares a series of hardcoded priorities which handlers can use.

Note: Priority is a misleading name, because higher "priority" means that the handler should execute *later* in the sequence and not earlier. The reasoning for this is that higher order handlers "get the last word" over lower order handlers. This naming may be changed in future PocketMine-MP versions.

List of priorities

At the time of writing, the following priorities exist:

Name	Description
LOWEST	These handlers will be executed first. This should be used when the original event state is needed (before later handlers modify it), or when the handler is making a low-priority modification (to allow other handlers to override it).
LOW	
NORMAL	If a priority is not specified, handlers will be registered here by default.
HIGH	
HIGHEST	Handlers registered at this priority get the final say in the event's outcome.
MONITOR	These handlers will execute last. **No modification to the event's outcome should be made at this priority, including cancellation**. This should be used to only for **monitoring the outcome** of an event.

Warning: When multiple handlers are registered to the same priority, the order of execution is undefined.

16.3 Inventory Transactions

Inventory transactions are a system used to group changes to inventories, dropped items or other item stack locations. This system is used to provide a coherent view of activity in inventories caused by players, and to prevent cheating or creating/destroying items in multiplayer races.

16.3.1 Validation

A standard inventory transaction is valid when the following conditions are met:

- The computed balance of the transaction is zero. This means:
  - Every input must be matched by an equal number of identical outputs.
  - No extra items can be created (outputs without a balancing input)
  - No items can be destroyed (inputs without a balancing output)
  - Item userdata may not change on any item (custom names, lore, etc)
- The transaction must have at least 1 action.
- All actions in the transaction must consider themselves valid. This includes checks such as validating that the input item stack for a slot change matches the existing stack in the inventory.

16.3.2 Inventory actions

An inventory action describes a single change to an item stack involved in the transaction. It has two primary components:

- Input (for validation): The item stack that should exist in the target location before the transaction takes place.
- Output (for execution): The item stack that will be set into the target location once the transaction is completed.

Note: A transaction is composed of an **unordered set** of inventory actions. The system is **explicitly designed NOT to care about ordering**, so your plugin should also not depend on any ordering.

Inventory action types

There are several types of inventory actions which provide different methods of modifying the balance of a transaction.

Slot change

A standard change in an inventory or container, e.g. a chest. The target location is a specific slot number in the inventory. To be valid, the input item for this action type must match the item in the inventory at the point of validation.

Drop item

An item stack dropped into the world as an entity. This has no validation criteria other than the usual transaction balancing requirements.

Creative inventory

An item stack being created or destroyed. This action type is only legal when the source player is in creative mode (and the item must exist in the creative inventory when creating items).

16.4 Plugin formats

PocketMine-MP supports several plugin formats. The standard ones are described below; you can make your own by making a custom plugin loader.

16.4.1 Standard plugin types

Phar

This plugin format consists of a single PHAR file which bundles all of the files necessary to make the plugin work. This type of plugin is commonly used to distribute pre-made plugins, because they are easy for users to download and move around without needing to modify the plugin code.

Loading

1. Drop the `.phar` file into your `plugins` directory.
2. Restart the server. The plugin will be loaded (if compatible).

16.4.2 Development plugin types

Since the standard types of plugins are usually pretty difficult to work with while developing a plugin, there are more types of plugins that developers can use, but these are **not** recommended for distribution.

Folder

Note: this plugin format is **not** enabled by default. The DevTools plugin is required to load this type of plugin.

This plugin format is similar to a PHAR plugin in structure, but all the files are in a folder on the disk instead of inside a phar file. This gives you easy access to the source code when you're developing a plugin.

This plugin format is commonly used for development, because it is just as feature-complete as a PHAR plugin, and can be compiled to a PHAR plugin when you're ready to release it.

You can see the ExamplePlugin or any plugin source-code repository to get an idea of how this should look.

Loading

1. If you don't have the DevTools plugin, download its phar file and put it in your `plugins` folder.
2. Move the folder containing the plugin's source code into your `plugins` folder. The plugin's folder should contain a `plugin.yml` file and a `src` folder.
3. Restart the server and the plugin will be loaded.

Script

This plugin format works out of the box, but it's not recommended for production use, and has less features than a normal plugin would. This format is useful if you want to quickly test some features but don't need a full-blown plugin.

See this thread to see how a script plugin looks and works.

Loading

1. Drop the *.php* file into your *plugins* folder.
2. Restart the server and the plugin will be loaded.

16.4.3 Frequently Asked Questions

Does the `/reload` command reload plugin source code?

No. This is not currently possible.

How do I load a `.zip` plugin?

PocketMine-MP does not directly support loading zip plugins, but there may be third-party plugins available which allow you to do this. However, this is not recommended.

16.5 Plugin manifest fields

A plugin manifest is a document which contains information about a plugin. It is usually found in a *plugin.yml* file, but this is not mandatory.

Contents

16.5.1 Required fields

name

Type: `string`

Name of the plugin. This may contain letters, numbers, hyphens, periods and underscores. It may also contain spaces, but this is discouraged.

version

Type: `string`

Self explanatory. It's recommended to use a 3-point semantic version, but this can be anything you like.

main

Type: `string`

Fully-qualified name of the main class. This class must meet the following criteria:

- MUST NOT be abstract
- MUST implement the `pocketmine\plugin\Plugin` interface

api

Type: `string` or `string[]`

The API version(s) that the plugin is compatible with. If the plugin's API version is not compatible with that of the server, the server will refuse to load the plugin. More info on API versioning can be found *here <api_version_spec>*.

16.5.2 Optional fields

Cosmetic

website

Type: `string`

Website for the plugin.

description

Type: `string`

Short description of the plugin.

prefix

Type: `string`

Alternative prefix to use in the plugin's log messages. Defaults to the plugin name.

author

Type: `string`

Author name of the plugin.

authors

Type: `string[]`

A list of author names, if there are more than one. If both `author` and `authors` are defined, a list will be formed containing both.

Plugin loading controls

load

Type: `string`

When in the startup sequence to prefer loading this plugin. Currently can be one of `STARTUP` or `POSTWORLD`. See plugin load order. (TODO: add a link here)

depend

Type: `string` or `string[]`

List of plugins that this plugin depends on. Plugin will not load if any of these plugins are missing.

softdepend

Type: `string` or `string[]`

List of plugins that the plugin can **optionally** depend on. Plugins in this list must load prior to the plugin soft-depending on them.

loadbefore

Type: `string` or `string[]`

List of plugins that this plugin must load prior to. Works like a soft-dependency in reverse.

extensions

Type: `array`

List of PHP extensions that the plugin requires. Plugin will not load if any are missing or have unmet version constraints. TODO: examples

mcpe-protocol

Type: `int` or `int[]`

List of Minecraft PE network protocol versions the plugin is compatible with. Plugin will fail to load if the current server protocol version is not in this list.

os

New in version 3.12.0.

Type: `string` or `string[]`

List of operating systems that the plugin will run on. If not present, the plugin will load on any OS. Possible values include `win`, `mac`, `linux`, `android`, `ios`, `bsd`.

Misc

commands

Type: `array`

Definitions of commands implemented by this plugin in the `onCommand()` of the `PluginBase`.

Example:

```
commands:
  # The name of the command the user will type to execute it
  example:
    # Description that will be shown in help command
    description: Example command
    # Shown to the user if they type the command in incorrectly
    usage: "/example"
    aliases:
      - ex
      - examp
    # Permission required to run the command
    permission: exampleperm.command.example
    # Shown to the user if they don't have permission to run the command
    permission-message: "You do not have permission to use this example command!"
```

permissions

Type: `array`

List of permissions defined by this plugin, usually used for commands.

Example:

```
permissions:
  exampleperm.command.example:
    description: "Allows the user to run the example command"
    # Default state of the permission. Explanation of each value:
    # op: only op players have this permission by default
    # true: everyone has this permission by default
    # false: no one has this permission by default
    default: true
```

src-namespace-prefix

New in version 4.0.0.

Type: `string`

Base namespace of the classes in your `src/` folder. Defaults to empty string.

This allows you to have a longer namespace for your classes without having to create useless nested folders in your plugin structure.

Examples:

Value of `src-namespace-prefix`	Name of class including namespace	Path class will be loaded from
(empty)	`YourName\PluginName\Main`	`src/YourName/PluginName/Main.php`
	`YourName\PluginName\SubNamespace\OtherClass`	`src/YourName/PluginName/SubNamespace/OtherClass.php`
`YourName\PluginName`	`YourName\PluginName\Main`	`src/Main.php`
	`YourName\PluginName\SubNamespace\OtherClass`	`src/SubNamespace/OtherClass.php`

16.6 Threading in PHP - WTF?

This page aims to give some technical insight into what's required to get threading in PHP, and why every threading extension for PHP sucks.

16.6.1 PHP is not designed for this!

PHP's design goal is to provide an easy scripting language for use on webservers, for delivering webpages to browsers. It's popular for this use case. Unfortunately, web requests don't usually need threading in user code, since web requests typically last a few seconds at most, and are mostly I/O bound, not needing the use of many CPU cores. PHP has flourished for over 20 years without support for user threads, and it's likely this will continue to be the case for years to come.

Because almost all use-cases for PHP involve serving web requests, the design choices made by the PHP developers over the decades have been oriented with a webserver-first approach. Serving webpages is something PHP does very well. One of these design choices has been to make no effort whatsoever to implement support for userland threading.

16.6.2 Almost everything must be copied

Every complex data structure in PHP is non-thread-safe. This applies to userland types such as arrays, objects, strings, and resources, and also applies to things you might not expect - functions, classes, and constants. Reference counts on these data structures are not atomic, and the Zend Engine's memory manager goes out of its way to prevent stuff from being shared between one thread's memory manager and another.

This means **all of these things must be copied** in order to get them from one thread to another, which makes passing large data from one thread to another very expensive, and therefore **severely limits** the viable use cases of PHP threading. The CPU cost of copying the required data onto the target thread can easily exceed the time saved by threading.

The only viable use cases are those which require relatively **little transfer of data** between threads but have relatively **large time cost**. Currently, PocketMine-MP only uses threads for world generation, light calculation, network compression, some internal network systems, and the occasional cURL request.

16.6.3 Threads don't inherit anything

Every new thread in a ZTS build of PHP gets a completely new interpreter context. This means that no user classes are loaded (unless preloaded by OPcache).

Classes and functions aren't shared (or shareable) between threads, and therefore must be copied, or otherwise reloaded, onto a new thread.

Code to copy class and function data structures from one thread to another makes up the majority of code in PHP threading extensions such as pthreads (and therefore the majority of the bugs).

Other extensions such as parallel reduce complexity by forcing the use of autoloaders to reload classes on new threads instead of copying them, but this imposes some limitations, since not all stuff can be autoloaded (e.g. anonymous classes). In addition, it still needs to be able to copy functions (since its unit of work is a closure).

To make matters worse, these internal data structures are subject to change from one PHP version to the next, meaning that this code often breaks, and is the main obstacle to upgrading PHP version in projects like PocketMine-MP.

16.6.4 ZTS (Zend Thread Safety)

The Zend Engine at the heart of the PHP interpreter provides two modes of operation.

- NTS (Non Thread Safe) makes up the vast majority of PHP installations. In this mode, there may only be one interpreter context in a process. Thread safety is not usually needed in a typical PHP use-case, since a webserver just spins up a new PHP process for each request.
- ZTS (Zend Thread Safe) is used to allow each webserver request to run in a new thread of the same process, rather than in a separate process. Each thread has its own independent interpreter context. This mode is typically used on Windows, where fork(2) is not available.

Neither of these modes is suitable for user threading. NTS is (obviously) not thread-safe, so accessing global state on different threads at the same time would lead to data races and possibly crashes.

ZTS is marginally less unsuitable. While ZTS enables running multiple independent threads of PHP code in the same process, it does so by making sure that each thread can't access any state from other threads. This is great for webservers, where different requests shouldn't be able to interfere with each other, but it's a big obstacle for userland threading, where interaction between different threads is necessary.

Every threading extension made for PHP has built on top of the ZTS mode, and from there done an enormous amount of hacks to make different threads able to interact with each other, despite the limitations imposed by the Zend Engine.

Implementing threading properly into PHP would require a significant amount of changes to the PHP core, which it seems no one in the world is inclined to do. Until the time comes when a knight in shining armour implements threading properly into PHP, we're stuck with stuff like pthreads and all the hacks necessary to make it even remotely usable.

CHAPTER

SEVENTEEN

EXTERNAL DEVELOPMENT RESOURCES

- Doxygen - API documentation for latest release
- DevTools - Development tools plugin for creating plugins
- ExamplePlugin - Example plugin demonstrating some basic API features