

# WaterdogPE Setup

Simple Guide through WaterdogPE setup process.

- [Starting Waterdog](#)
- [Proxy Configuration](#)
- [Software Compatibility](#)
- [Troubleshooting](#)

# Starting Waterdog

To start pre-compiled WaterdogPE (WDPE) you will need Java 11 installed. If you want to start proxy with older version of Java, you must compile WDPE yourself.

1. Download the latest built `Waterdog.jar` from our jenkins server.
2. Place the file inside a new directory dedicated to Waterdog.
3. Create a new startup script to launch the the JAR.
4. Once you have successfully installed WDPE, it is time to get it working properly. One of the most essential steps is to configure your downstream server instances (Nukkit, PMMP) to run in offline-mode, which can be achieved by modifying `server.properties`. You would need to set `xbox-auth` to `false`.
5. PocketMine-MP has added another option which must be disabled when using proxy. Since server thinks player is unauthenticated, his XUID will be unset. PMMP implements new security check where it verifies if players last XUID match the current one. In proxy case it will *not* match and player will be disconnected. To solve this issue, please disable `player.verify-xuid` in `pocketmine.yml` file.

## Startup script

When using Java 11 we recommend adding this parameters, which will add support for some netty features.

```
-Dio.netty.tryReflectionSetAccessible=true
--add-opens java.base/jdk.internal.misc=ALL-UNNAMED
```

Minimum allocated memory can be set using `-Xms<size>`, maximum `-Xmx<size>`.

## Windows

In dedicated directory create `start.bat` file with following code:

```
@echo off
java -Xms512M -Xmx4G -jar Waterdog.jar
pause
```

## Linux

For Linux it is recommended to start proxy in bash environment. Create `start.sh` file with following code:

```
#!/bin/bash
java -Xms512M -Xmx4G -jar Waterdog.jar
```

While using above code in your startup scripts is a template, you can also replace **512M** with the amount of **initial memory pool** that you'd like to allocate and **4G** with the amount of **maximum memory pool** that you'd like to allocate to the proxy. This is completely dependent on the underlying hardware you use and on your needs. You don't have to do this if you don't want to.

To give your script executable permissions `chmod a+x start.sh` To start proxy you can execute from terminal `bash start.sh`.

# Proxy Configuration

WaterdogPE uses `config.yml` file to store all general settings which will be loaded on startup.

Default messages are stored in `lang.ini` file.

## Configuration

Configuration is already commented inside of the file so lets point out only some details.

## Login Extras

Login extras are used to pass modified, added variables in client data. Some softwares does not like adding custom attributes to client skin or chain data. If `use_login_extras` is disabled proxy won't add any extra attributes.

## Ip Forward

One of the extra attributes is `Waterdog_IP`. Because connection with downstream is initialized by proxy, downstream server thinks that address if proxy is the address of client. This attribute is used to send original address of upstream.

## Prefer Fast Transfer

Proxy has the ability to handle `TransferServer` packet sent by downstream and use its address and port to determine new downstream. If this option is enabled and new downstream was found WDPE will transfer player without disconnecting from proxy (using "fast" transfer).

## Fast Codec

Because proxy is bridge between client and downstream every packet has to pass through it. When lot of clients are connected it may be resource-intensive to decode and then encode every

packet. Enabling `use_fast_codec` option will load customized protocol codec which will pass original data of packets that are not handled by proxy.

**This feature may cause problems with some plugins!** If your plugin is using any packet which is not registered in customized codec, it will throw an exception. For those who would like to use fast codec and send other packets we have created event which will be called before codec registration. Please refer to code documentation.

## Compression Settings

Data sent between client, downstream and proxy are compressed to save bandwidth. But choosing wrong, too high compression level may be CPU intensive.

Higher level = more CPU, less bandwidth usage. Set to 0 to no compression, 9 to highest compression ratio.

### Downstream Compression

Compression between downstream and proxy could be usually disabled or at least lowered. Proxy should have fast enough connection to downstream server. Use `downstream_compression_level` to set proxy to downstream compression level.

### Upstream compression

Some clients may have slower connection to your server therefore it is recommended to compress data. Do not set compression level too high or performance may drop. Use

`downstream_compression_level` to set proxy to client compression level.

## Education Edition mode

If downstream server has enabled education features, `enable_edu_features` option should be enabled or game may start crashing.

When this option is enabled, additional education resource pack is applied in order to load required education resources. **Note that this resources can be enforced only when**

`enable_packs` **option is enabled.**

# Resource Pack Caching

When client downloads resource pack its data are sent in chunks to prevent sending large payloads. These chunks are created from resource pack file. Every client can request different part of file (different chunk). Therefore chunk is every time loaded from original file. This may cause higher disk IO with lot of joining clients. To improve performance, data of resource pack can be cached. **Caching big resource packs will use more RAM.** Use `pack_cache_size` to limit maximum pack size (in MB) to be cached. If size of pack is bigger, it won't be cached.

## Idle Threads

WaterdogPE uses threaded executors for scheduler and asynchronous event executing. Executor will destroy unused idle threads. You can specify count of idle threads which will not be destroyed using `default_idle_threads`. Set to `-1` to auto-detect idle threads count by CPU cores.

## Example config:

```
# Waterdog Main Configuration file
# Configure your desired network settings here.

# A list of all downstream servers that are available right after starting
# address field is formatted using ip:port
# publicAddress is optional and can be set to the ip players can directly connect through
servers:
  lobby1:
    address: 127.0.0.1:19133
    public_address: play.myserver.com:19133
listener:
  # The Motd which will be displayed in the server tab of a player and returned during ping
  motd: $bWaterdog$3PE
  # The server priority list. If not changed by plugins, the proxy will connect the player to
  the first of those servers
  priorities:
    - lobby1
```

```
# The address to bind the server to
host: 0.0.0.0:19132

# The maximum amount of players that can connect to this proxy instance
max_players: 20

# Map the ip a player joined through to a specific server
# for example skywars.xyz.com => SkyWars-1
# when a player connects using skywars-xyz.com as the serverIp, he will be connected to
SkyWars-1 directly
forced_hosts: {}

# Case-Sensitive permission list for players (empty using {})
permissions:
  TobiasDev:
    - waterdog.player.transfer
    - waterdog.player.list
  alemiz003:
    - waterdog.player.transfer
    - waterdog.player.list

# List of permissions each player should get by default (empty using [])
permissions_default:
  - waterdog.command.help
  - waterdog.command.info

# Whether the debug output in the console should be enabled or not
enable_debug: false

# If enabled, encrypted connection between client and proxy will be created
upstream_encryption: true

# If enabled, only players which are authenticated with XBOX Live can join. If disabled,
anyone can connect *with any name*
online_mode: true

# If enabled, the proxy will be able to bind to an Ipv6 Address
enable_ipv6: false

# If enabled, the proxy will pass information like XUID or IP to the downstream server using
custom fields in the LoginPacket
use_login_extras: true

# Replaces username spaces with underscores if enabled
replace_username_spaces: false

# Whether server query should be enabled
enable_query: true

# If enabled, when receiving a McpeTransferPacket, the proxy will check if the target server
is in the downstream list, and if yes, use the fast transfer mechanism
```

```
prefer_fast_transfer: true
# Fast-codec only decodes the packets required by the proxy, everything else will be passed
rawly. Disabling this can create a performance hit
use_fast_codec: true
# If enabled, the proxy will inject all the proxy commands in the AvailableCommandsPacket,
enabling autocompletion
inject_proxy_commands: true
# Upstream server compression ratio(proxy to client), higher = less bandwidth, more cpu,
lower vice versa
upstream_compression_level: 6
# Upstream server compression ratio(proxy to downstream server), higher = less bandwidth,
more cpu, lower vice versa
downstream_compression_level: 2
# Education features require small adjustments to work correctly. Enable this option if any
of downstream servers support education features.
enable_edu_features: false
# Enable/Disable the resource pack system
enable_packs: true
# Whether texture packs should be enforced
force_apply_packs: false
# You can set maximum pack size in MB to be cached.
pack_cache_size: 16
# Creating threads may be in some situations expensive. Specify minimum count of idle threads
per internal thread executors. Set to -1 to auto-detect by core count.
default_idle_threads: -1
```

# Language Settings

Default messages can be edited in `lang.ini` file.

## Format

Proxy uses TranslationContainer to translate translation key to real message. It replaces patterns ( `{%0}`, `{%1}`, ... ) with pre-assigned values.

### Example:



waterdog.query.start=Started query on address {%0}

# Software Compatibility

WaterdogPE is currently not compatible with every server software. This table should show which ones are generally supported and which ones are not.

software	Compatible
Nukkit	
- Vanilla	yes
PocketMine API 4	yes
PocketMine API 5	yes
Dragonfly	yes
MINET	yes
GoMint	yes
Jukebox	yes
PowerNukkit	no
PowerNukkitX	no
Bedrock Dedicated Server	may work

# Troubleshooting

This page covers errors casually reported by users and how to resolve them.

## Setup

### Access proxy running on localhost

When binding the proxy to localhost, windows enforces program restrictions preventing minecraft from accessing the server. These restrictions can be lifted by running the following command *in an administrator shell*

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

### Wrong java version

The following error can be observed if you start WaterdogPE with the wrong java version:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: dev/waterdog/WaterdogPE has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0
```

This happens when you try to run WaterdogPE using Java 8, meanwhile WaterdogPE is built using Java 11. To resolve this, run it using Java 11.

### JSON property "Waterdog\_XUID" does not exist

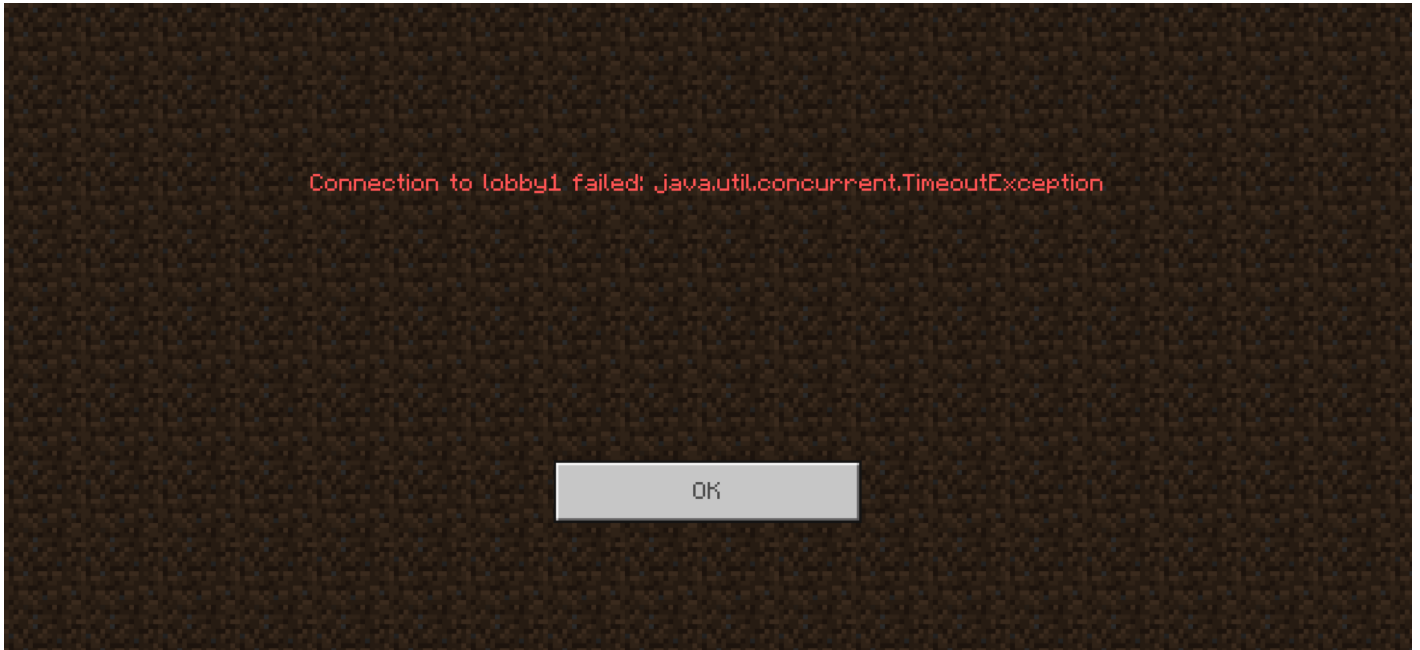
WaterdogPE passes down information like XUID and IP to the downstream server using custom fields in the Login Packet. If you are using any version of Pocketmine (API 3 or API 4), you may encounter the above error.

In order to resolve this issue, you should disable the property "use\_login\_extras" in your WaterdogPE configuration file `config.yml` to match:

```
use_login_extras: false
```

# java.util.concurrent.TimeoutException

When you get kicked for a screen like this:



Likelihood is extremely high that this is a configuration error. It means that the proxy is not able to communicate with the game server you're trying to join. This can most commonly be a firewall issue, configuration mistake or something along those lines. Before you report this as an error or request support, make sure to check the following boxes:

- Are the IP and port in the config.yml correct?
- Is the server running the latest version of the config.yml?
- Can I join the server with the same credentials directly through Minecraft?
- Is there any message on the game server about the player trying to join (turn on debug)
- Can I ping the target machine?

If all of these questions did not help you find the issue, then you can ask for support on our support discord.

# "You need to authenticate to Microsoft services"

This issue usually occurs when a user is either not logged into Xbox Live and the proxy has online-mode enabled or the downstream server has online-mode enabled.

To resolve this, make sure you are signed in to Xbox Live in Minecraft and that all your downstream servers have the "xbox-auth" option disabled.

## "XUID does not match"

Because of a security vulnerability which users could use to join with the same Xbox username like an already existing one, server softwares like PocketMine-MP compare the XUID to the account name. An account won't be able to join if the XUID mismatches with a previous XUID.

To resolve this issue, delete the .dat file of that player or set the "player.verify-xuid" option in your pocketmine.yml to false.

That should look like this:

```
player:
  verify-xuid: false
```

## Resource pack manifest.json is invalid or was not found

This error means that your resource pack was not properly formatted.

“ It is important that you zip the content of the pack, not the folder.

That means, when you have your resource pack, you select all the files in the folder of the manifest.json and zip those files. You **do not** zip the folder that contains the manifest.json.