

# StarGate Plugins

Guide thought plugin configuration and usage.

- [Server Setup](#)
- [Client Setup](#)

# Server Setup

Configuring and starting default StarGate server shouldn't be problem. StarGate server comes as plugin for Bungeecord (and its forks) and for WaterdogPE. Both plugins use similar or same API methods and use same config file.

## Server Plugin

StarGate server is used to handle new pending connections, route/handle incoming packets from the clients. Service uses `tcp` sockets, by default port `47007` is used.

You should include `StarGate` plugin into dependencies of your plugin inside of `plugin.yml` file.

Your plugin will be then enabled after server plugin.

## Config

- StarGate uses simple authentication based on string password. You can adjust security level by implementing the extra layer of authentication using custom packets.
- When option `blockSameNames` is enabled and client with same name name included in `HandshakeData`, it will be disconnected.
- If `debug` is enabled `StarGateLogger#debug()` messages will be shown in console.

This is how default config should look like.

```
# On this port StarGate server will listen for new connections.
serverPort: 47007
# To make connection secure enough set strong password that will be used to authenticate
clients.
password: "123456789"
# Clients should not have same name and should be same as name of downstream server.
# Disabling this option will allow to join more clients using same client name.
# If enabled, some packets that use client name to identify client, might not work.
blockSameNames: true
```

```
# Enable debug logger
debug: true
```

## Server Events

Plugins can use events to handle new connected, authenticated or disconnected session. Here is simple overview of currently available events.

- `ClientConnectedEvent` : Called once new session is created. At this point session is NOT authenticated and will not accept other than `HandshakePacket` .
- `ClientAuthenticatedEvent` : Called once session is successfully authenticated using simple password. You can set custom packet handler to session when this event is called. *WaterdogPE plugin marks this event as `@AsyncEvent` .*
- `ClientDisconnectedEvent` : Called once session has been disconnected or closed. *WaterdogPE plugin marks this event as `@AsyncEvent` .*

## Registering packets

Assuming your plugin is enabled after StarGate server plugin, you can register custom packets inside of `onEnable()` method.

```
ProtocolCodec codec = StarGate.getInstance().getServer().getProtocolCodec();
codec.registerPacket(StarGatePackets.SERVER_INFO_REQUEST_PACKET,
    ServerInfoRequestPacket.class);
```

## Implementing Own Server

To create new server instance you should create new class which will implement `ServerLoader` .

Currently, it is used only to provide own `StarGateLogger` implementation. Using public

```
StarGateServer(bindAddress : InetSocketAddress, password : String, loader : ServerLoader)
```

constructor we create new server instance. To start server we use `StarGateServer#start()` .

Simple example:

```
MyServerLoader loader = new MyServerLoader();
```

```
InetSocketAddress address = new InetSocketAddress("0.0.0.0", 47007);
StarGateServer server = new StarGateServer(address, "12345", loader);
// You can register custom packets here
server.start();
```

# Client Setup

Client is used to communicate with StarGate server. In Minecraft environment, we know clients as plugin for server. Stargate comes with default implementation of clients for:

- PocketMine-MP 3 (StarGate-Atlantis, PHP)
- NukkitX (StarGate-Universe, Java)

## Plugin Configuration

**Server clients use same configuration file:**

- `connections` map is used to represent the map of connections. One plugin can create several clients and connect to more different servers.
- If debug is enabled `debug` messages will be shown in console.
- Each packet can have different level of logging. Packets which have equal or higher log level value `logLevel` option will be logged in console.
- When `autoStart` option is enabled all clients will be started and try to connect once plugin enables.
- Option `tickInterval` is the interval in ticks between each tick when client will process handled buffer and decode packets. The smaller interval will be given, the faster will packets be handled. **This option is used only in StarGate-Atlantis plugin.**

```
# Supports more connections
connections:
  lobby1:
    address: 192.168.0.50
    port: 47007
    password: 123456789

# Enable debug logger
debug: true
```

```
# Change log level to log packets in debug mode.
logLevel: 0
# Default client will be used if no clientName specified in api functions
defaultClient: "lobby1"
# Start clients after plugin is enabled.
autoStart: true
# Interval in ticks to receive data from connection
tickInterval: 2
```

# Session Events

- `ClientCreationEvent` : Called once new StarGate Client is created (usually when StarGate-Universe plugin is enabled). It can be used to adjust ProtocolCodec. *Note that if your plugin is loaded after StarGate-Universe your event listener won't handle this event.*
- `ClientConnectedEvent` : Called once new session is created. At this point session is NOT authenticated.
- `ClientAuthenticatedEvent` : Called once session is successfully authenticated using simple password ( `ServerHandshakePacket` is received). You can set custom packet handler to session when this event is called.
- `ClientDisconnectedEvent` : Called once session has been disconnected or closed.

# Plugin Methods

Plugins for all platforms expose these common methods:

- `getClient(String) : StarGateClient` returns the client matched by name.
- `getDefaultClient() : StarGateClient` returns default client as set in config.
- `getClientsCopy() : StarGateClient[]` returns list of currently created clients. Modifying this list will not modify an original list. Method is called `getClients()` in StarGate-Atlantis plugin.
- `setLogLevel(int)` method is used to change log level of all clients.
- `getLogLevel() : int` returns currently used log level.
- `transferPlayer(Player, targetServer, clientName)` is supposed to send `TransferPacket` to StarGate server (and transfer player to another downstream). When `clientName` is null,

default client will be used. If `clientName` does not match any client session, packet won't be sent.

- `serverInfo(serverName, selfMode, clientName : CompletableFuture)` method is used to retrieve information of another downstream server. If `selfMode = true` response data of whole proxy will be rest beack. If `clientName` does not match any client session, packet won't be sent, null will be returned. Method returns future which is completed once response if received. When using StarGate-Atlantis `PacketResponse` is returned. *Note that some StarGate server implementations might not support this method.*

# StarGate-Universe

StarGate Universe contains Java client implementations. Currently supporting only NukkitX client.

## NukkitX

Use Maven dependency in your plugin (see below). You should probably include StarGate Common dependency aswell.

```
<dependency>
  <groupId>alemiz.sgu.nukkit</groupId>
  <artifactId>sgu-nukkit</artifactId>
  <version>2.1-SNAPSHOT</version>
  <scope>provided</scope>
</dependency>
```

You should also include `StarGate-Universe` in `plugin.yml` dependencies list.

Latest build can be found on Waterdog [jenkins server](#), artifacts are uploaded to Waterdog repository server.

To access plugin instance we can use `StarGateUniverse plugin = StarGate.getInstance()`.

If you have disabled `autoStart` option in config, you can start all clients like below:

```
StarGateUniverse plugin = StarGate.getInstance()
for (StarGateClient client : plugin.getClientsCopy()){
```

```
// You can adjust ProtocolCodec here
client.start();
}
```

# StarGate-Atlantis

StarGate Atlantis is client implementation based on PHP created for PocketMine-MP (currently version 3 only). To make your IDE index plugin classes you can use `phar` file obtained from [Poggit](#) and include it as dependency.

To access plugin instance we can use `$plugin = StarGateAtlantis::getInstance()`.

If you have disabled `autoStart` option in config, you can start all clients like below:

```
$plugin = StarGateAtlantis::getInstance()
foreach ($plugin->getClients() as $client){
    // You can adjust ProtocolCodec here
    $client->connect();
}
```