
SpockBot Documentation

Release 0.1.2

The SpockBot Project

May 01, 2016

1	Welcome to SpockBot's documentation!	1
2	Indices and tables	47
	Python Module Index	49

Welcome to SpockBot's documentation!

1.1 Installation

If you are looking to contribute to SpockBot please follow the instructions [here](#).

1. Install cryptography first, instructions for that are [here](#)
2. Clone the repository locally:

```
git clone https://github.com/SpockBotMC/SpockBot.git
```

3. Install SpockBot:

```
cd SpockBot
python3 setup.py install
```

1.2 Getting Started

See the [Basic Example Bot](#) for basic usage of SpockBot.

Also see the [Extra Repository](#) for some more plugin examples.

1.3 Plugins

1.3.1 Core Plugins

Auth

Description

Provides authorization functions for Mojang's login and session servers

Events

Undocumented

Methods and Attributes

`class spockbot.plugins.core.auth.AuthCore(event, online_mode, auth_timeout)`

```
auth_token
client_token
get_shared_secret()
get_username()
password
send_session_auth(pubkey_raw, server_id_raw)
set_auth_token(auth_token)
set_client_token(client_token)
set_password(password)
set_username(username)
shared_secret
start_session()
username
```

Event

Description

Provides the core event loop

Events

event_start

Fired before the first iteration of the event loop. The default StartPlugin depends on this event.

event_tick

Fired at the beginning of every iteration of the event loop. The default NetPlugin depends on this event.

event_kill

Fired after the last iteration of the event loop. Many plugins depend on this event for environment clean up purposes.

Methods and Attributes

Net

Description

Provides an asynchronous, crypto and compression aware socket for connecting to servers and processing incoming packet data. Coordinates with the Timers plugin to honor wall-clock timers

Events

Undocumented

Methods and Attributes

class `spockbot.plugins.core.net.NetCore` (*sock, event, select*)

connect (*host='localhost', port=25565*)

disable_crypto ()

enable_crypto (*secret_key*)

push (*packet*)

push_packet (*ident, data*)

read_packet (*data=b''*)

reset (*sock*)

set_comp_state (*threshold*)

set_proto_state (*state*)

Settings

Description

Events

Undocumented

Methods and Attributes

Undocumented

Task Manager

Description

Events

Undocumented

Methods and Attributes

Undocumented

Ticker

Description

Registers timers to provide the necessary tick rates expected by MC servers

Events

action_tick

This tick is for scheduling physical actions for the client, specifically those related to movement. Any plugin that wishes to move the client should schedule those movements by hooking into *action_tick* and calling the appropriate pathfinding or movement methods.

physics_tick

This tick fires immediately after *action_tick* and signals that all movement actions have been scheduled and the client is ready to process the movement actions as well as world forces to resolve a position for the client. The default PhysicsPlugin depends on this event.

client_tick

This tick fires immediately after *physics_tick* and signals that a new position has been resolved for the client and is ready to be sent to the server. The default MovementPlugin depends on this event.

Timer

Description

Events

Undocumented

Methods and Attributes

- *Auth* - Handling Minecraft account authentication
- *Event* - Provides an event system
- *Net* - Handles packets
- *Settings* - Allows custom settings handlers
- *Task Manager* - Create state machine tasks
- *Ticker* - Ticks 20 times a second
- *Timer* - Create timers to do stuff later

1.3.2 Helper Plugins

Channels

Description

Provides interface for Plugin Channels

Events

Undocumented

Methods and Attributes

class `spockbot.plugins.helpers.channels.ChannelsCore` (*net*)

decode (*structure, data*)

encode (*structure, data*)

send (*channel, data*)

Send a plugin channel message

Chat

Description

Processes chat into easy to use events

Events

chat

Chat event was recieved

Payload

```
{'position': position, 'raw': json_data, 'text': text, 'type': chat_type, 'message': msg, 'name': name}
```

position

int

Where the text should be displayed

raw

dict

Raw json data from the chat message

text

string

The text of the chat message

type

string

The type of message (achievement, admin, announcement, emote, incoming, outgoing, text)

message

string

message

uuid

string

UUID of the player who sent the message

Methods and Attributes

class `spockbot.plugins.helpers.chat.ChatCore` (*net*)

chat (*message*)

whisper (*player, message*)

ClientInfo

Description

ClientInfo is a central plugin for recording data about the client, e.g. Health, position, and some auxillary information like the player list. Plugins subscribing to ClientInfo's events don't have to independently track this information on their own.

Events

client_login_success

Client Info is done processing the Login Success packet

client_join_game

Client Info is done processing the Join Game packet

Payload

<code>GameInfo()</code>

client_spawn_update

Client Info is done processing the Spawn Position packet

Payload

<code>Position()</code>

client_health_update

When the players health changes

Payload

<code>PlayerHealth()</code>

client_death

When the player health becomes 0.0

Payload

<code>PlayerHealth()</code>

client_position_update

When the player is moved by the server

Payload

PlayerPosition()

client_add_player

Player added to the player list

Payload

PlayerListItem()

client_update_player

Player info is updated on the player list

Payload

PlayerListItem()

client_remove_player

Player removed from the player list

Payload

PlayerListItem()

Methods and Attributes

class spockbot.plugins.helpers.clientinfo.**ClientInfo**

eid

int

Entity ID of the player

name

str

Player's Username

uuid

str

Player's UUID

abilities

Abilities

Player's current movement state and speed

game_info

GameInfo

Information about the current world/server

spawn_position

Position

Players initial position

health

PlayerHealth

Player's health, food and saturation

position

PlayerPosition

Player's current position

player_list

dict

List of all players in the server

eye_pos

PlayerPosition

Player's eye position

eye_pos**reset ()**

Resets the information in ClientInfo

Craft

Description

Craft items.

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.craft.CraftPlugin (ploader, settings)
```

```
craft (item=None, meta=None, amount=1, recipe=None, parent=None)
```

Starts a `craft_task`. Either `item` or `recipe` has to be given.

Returns The recipe used for crafting.

Return type Optional[Recipe]

```
craft_task (recipe, amount=1)
```

A task that crafts `amount` items with `recipe`.

```
pl_announce = ('Craft',)
```

```
requires = ('Inventory', 'TaskManager')
```

Entities

Description

An entity tracker

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.entities.EntitiesCore
```

Interact

Description

Interact with the world: - swing the arm, sneak, sprint, jump with a horse, leave the bed - look around - dig/place/use blocks - use the held (active) item - use/attack entities - steer vehicles - place and write signs - edit and sign books

By default, the client sends swing and look packets like the vanilla client. This can be disabled by setting the `auto_swing` and `auto_look` flags.

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.interact.InteractPlugin (ploader, settings)
```

```
activate_item()
```

Use (hold right-click) the item in the active slot. Examples: pull the bow, start eating once, throw an egg.

```
attack_entity (entity)
```

```
cancel_digging()
```

```
click_block (pos, look_at_block=True, swing=True, **kwargs)
```

Click on a block. Examples: push button, open window, make redstone ore glow

Parameters

- **face** (*int*) – side of the block on which the block is placed on
- **cursor_pos** (*Vector3*) – where to click inside the block, each dimension 0-15

```
deactivate_item()
```

Stop using (release right-click) the item in the active slot. Examples: shoot the bow, stop eating.

```
dig_block (pos)
```

Not cancelable.

edit_book (*pages*)
Set the pages of current book in hand

finish_digging ()

jump_horse (*jump_boost=100*)

jump_vehicle ()

leave_bed ()

look (*yaw=0.0, pitch=0.0, radians=False*)

look_at (*pos*)

look_at_rel (*delta*)

look_rel (*d_yaw=0.0, d_pitch=0.0, radians=False*)

mount_vehicle (*entity*)

open_inventory ()

pl_announce = ('Interact',)

place_block (*pos, sneak=True, **kwargs*)
Place a block next to *pos*. If the block at *pos* is air, place at *pos*.

place_sign (*pos, lines=[], **place_block_kwargs*)
Place a sign block and write on it.

requires = ('ClientInfo', 'Event', 'Inventory', 'Net', 'Channels')

sign_book (*author, title*)
Sign current book in hand

sneak (*sneak=True*)

sprint (*sprint=True*)

start_digging (*pos, face=1*)

steer_vehicle (*left=0.0, forward=0.0, jump=False, unmount=False*)

swing_arm ()

unmount_vehicle ()

unsneak ()

unsprint ()

use_bucket (*pos*)
Using buckets is different from placing blocks. See "Special note on using buckets" in http://wiki.vg/Protocol#Player_Block_Placement

use_entity (*entity, cursor_pos=None, action=0*)
Uses (right-click) an entity to open its window. Setting *cursor_pos* sets *action* to "interact at".

write_book (*text, author='', title='', sign=False*)
Write text to the current book in hand, optionally sign the book

Inventory

Description

The Inventory plugin keeps track of the inventory and provides simple inventory analysis and manipulation.

Events

Undocumented

Methods and Attributes

class `spockbot.plugins.helpers.inventory.InventoryCore` (*net_plugin*, *send_click*)

Handles operations with the player inventory.

active_slot

click_slot (*slot*, *right=False*)

Left-click or right-click the slot.

Parameters *slot* (*Slot*) – The clicked slot. Can be *Slot* instance or integer. Set to `inventory.cursor_slot` for clicking outside the window.

close_window ()

creative_set_slot (*slot_nr=None*, *slot_dict=None*, *slot=None*)

drop_slot (*slot=None*, *drop_stack=False*)

Drop one or all items of the slot.

Does not wait for confirmation from the server. If you want that, use a `Task` and `yield inventory.async.drop_slot()` instead.

If *slot* is `None`, drops the `cursor_slot` or, if that's empty, the currently held item (`active_slot`).

Parameters *slot* (*Optional[Slot]*) – The dropped slot. Can be `None`, integer, or *Slot* instance.

Returns The action ID of the click

Return type `int`

find_slot (*wanted*, *slots=None*)

Searches the given slots or, if not given, active hotbar slot, hotbar, inventory, open window in this order.

Parameters *wanted* – function(*Slot*) or *Slot* or *itemID* or (*itemID*, *metadata*)

Returns The first slot containing the item or `None` if not found.

Return type `Optional[Slot]`

find_slots (*wanted*, *slots=None*)

Yields all slots containing the item. Searches the given slots or, if not given, active hotbar slot, hotbar, inventory, open window in this order.

Parameters *wanted* – function(*Slot*) or *Slot* or *itemID* or (*itemID*, *metadata*)

inv_slots_preferred

List of all available inventory slots in the preferred search order. Does not include the additional slots from the open window.

- 1.active slot
- 2.remainder of the hotbar
- 3.remainder of the persistent inventory

select_active_slot (*slot_or_hotbar_index*)

total_stored (*wanted, slots=None*)

Calculates the total number of items of that type in the current window or given slot range.

Parameters wanted – function(Slot) or Slot or itemID or (itemID, metadata)

Keep Alive

Description

Events

Undocumented

Methods and Attributes

Undocumented

Movement

Description

MovementPlugin provides a centralized plugin for controlling client movement so the client doesn't try to pull itself in a dozen directions.

Events

Undocumented

Methods and Attributes

class spockbot.plugins.helpers.movement.**MovementCore** (*plug*)

current_path

current_target

final_target

is_moving

stop ()

Pathfinding

Description

Very rough asynchronous pathfinding plugin Implements the Lazy Theta* pathfinding algorithm

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.pathfinding.PathfindingCore (start_path)
```

Physics

Description

A Physics module built from clean-rooming the Notchian Minecraft client

Collision detection and resolution is done by a Separating Axis Theorem implementation for concave shapes decomposed into Axis-Aligned Bounding Boxes. This isn't totally equivalent to vanilla behavior, but it's faster and Close Enough^TM

AKA this file does Minecraft physics

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.physics.PhysicsCore (pos, vec, abilities)
```

```
    jump ()  
    move_angle (angle, radians=False)  
    move_target (vector)  
    move_vector (vector)  
    sprint ()  
    walk ()
```

Respawn

Description

Events

Undocumented

Methods and Attributes

Undocumented

Start

Description

This plugin creates a convenient `start()` method and attaches it directly to the client. More complex bots will likely want to create their own initialization plugin, so `StartPlugin` stays out of the way unless you call the `start()` method. However, the `start()` method is very convenient for demos and tutorials, and illustrates the basic steps for initializing a bot.

Events

Undocumented

Methods and Attributes

Undocumented

World

Description

Provides a very raw (but very fast) world map for use by plugins. Plugins interested in a more comprehensive world map view can use `mcp.mapdata` to interpret blocks and their metadata more comprehensively. Planned to provide light level interpretation based on sky light and time of day

Events

Undocumented

Methods and Attributes

```
class spockbot.plugins.helpers.world.WorldData (dimension=0)
```

```
    new_dimension (dimension)
```

reset ()

update_time (*data*)

- *Channels* - Process Plugin Channels
- *Chat* - Process chat into simple events
- *ClientInfo* - Keep track of information about the client/player
- *Craft* - Craft items
- *Entities* - Keep track of entities
- *Interact* - Interact with the world
- *Inventory* - Interact with inventories in Minecraft
- *Keep Alive* - Keep sending the keep alive packet
- *Movement* - Basic movement around the world
- *Pathfinding* - Pathfinding
- *Physics* - Reimplmentation of the Vanilla physics
- *Respawn* - Auto respawn the player on death
- *Start* - Helper start system for doing handshake and login
- *World* - Keep track of world data

1.4 Modules

1.4.1 spockbot package

Subpackages

spockbot.mcdata package

Submodules

spockbot.mcdata.biomes module

`spockbot.mcdata.biomes.get_biome` (*biome*)

spockbot.mcdata.blocks module

class `spockbot.mcdata.blocks.Block` (*meta=None*)

Bases: `object`

bounding_box = `None`

diggable = `True`

display_name = `'Block'`

drops = `[]`

hardness = `0.0`

harvest_tools = `[]`

```
id = -1
material = None
name = 'block'
slipperiness = 0.6
stack_size = 0
variations = {}
spockbot.mcdata.blocks.block_ext(*block_ids)
spockbot.mcdata.blocks.get_block(block, meta=0, init=True)
```

spockbot.mcdata.constants module These constants are used in some plugins, but do not belong there. Some of them can later be extracted from minecraft-data

spockbot.mcdata.items module

```
class spockbot.mcdata.items.Item(meta=None)
    Bases: object
    display_name = 'Item'
    id = -1
    name = 'item'
    stack_size = 0
    variations = {}
spockbot.mcdata.items.get_item(item, meta=0, init=True)
```

spockbot.mcdata.materials module

```
spockbot.mcdata.materials.get_material(name)
```

spockbot.mcdata.recipes module

```
class spockbot.mcdata.recipes.Recipe(raw)
    Bases: object
    ingredient_positions
        Returns In the form { (item_id, metadata) -> [(x, y, amount), ...] }
        Return type dict
    total_ingredient_amounts
        Returns In the form { (item_id, metadata) -> amount }
        Return type dict
class spockbot.mcdata.recipes.RecipeItem(id, meta, amount)
    Bases: tuple
    __getnewargs__()
        Return self as a plain tuple. Used by copy and pickle.
    __getstate__()
        Exclude the OrderedDict from pickling
```

static `__new__` (*_cls, id, meta, amount*)
 Create new instance of RecipeItem(id, meta, amount)

__repr__ ()
 Return a nicely formatted representation string

amount
 Alias for field number 2

id
 Alias for field number 0

meta
 Alias for field number 1

`spockbot.mdata.recipes.get_any_recipe` (*item, meta=None*)

`spockbot.mdata.recipes.iter_recipes` (*item_id, meta=None*)

`spockbot.mdata.recipes.reformat_item` (*raw, default_meta=None*)

`spockbot.mdata.recipes.reformat_shape` (*shape*)

spockbot.mdata.utils module

class `spockbot.mdata.utils.BoundingBox` (*w, h, d=None*)
 Bases: `spockbot.vector.Vector3`

class `spockbot.mdata.utils.Info`
 Bases: `object`

get_dict ()

set_dict (*data*)

`spockbot.mdata.utils.camel_case` (*text*)

`spockbot.mdata.utils.clean_var` (*text*)
 Turn text into a valid python classname or variable

`spockbot.mdata.utils.find_by` (*key, *args*)

`spockbot.mdata.utils.snake_case` (*text*)

`spockbot.mdata.utils.split_words` (*text*)

spockbot.mdata.windows module

class `spockbot.mdata.windows.BaseClick`
 Bases: `object`

apply (*inv_plugin*)
 Called by on_success(). Abstract method.

Parameters *inv_plugin* (`InventoryPlugin`) – inventory plugin instance

cleanup_if_empty (*slot*)

copy_slot_type (*slot_from, slot_to*)

get_packet (*inv_plugin*)
 Called by send_click() to prepare the sent packet. Abstract method.

Parameters *inv_plugin* (`InventoryPlugin`) – inventory plugin instance

mark_dirty (*slot*)

on_success (*inv_plugin*, *emit_set_slot*)

Called when the click was successful and should be applied to the inventory.

Parameters

- **inv_plugin** (*InventoryPlugin*) – inventory plugin instance
- **emit_set_slot** (*func*) – function to signal a slot change, should be `InventoryPlugin().emit_set_slot`

swap_slots (*slot_a*, *slot_b*)

transfer (*from_slot*, *to_slot*, *max_amount*)

class `spockbot.mdata.windows.DropClick` (*slot*, *drop_stack=False*)

Bases: `spockbot.mdata.windows.BaseClick`

apply (*inv_plugin*)

get_packet (*inv_plugin*)

class `spockbot.mdata.windows.SingleClick` (*slot*, *button=0*)

Bases: `spockbot.mdata.windows.BaseClick`

apply (*inv_plugin*)

get_packet (*inv_plugin*)

class `spockbot.mdata.windows.Slot` (*window*, *slot_nr*, *id=-1*, *damage=0*, *amount=0*, *enchants=None*)

Bases: object

copy ()

get_dict ()

Formats the slot for network packing.

is_empty

matches (*other*)

move_to_window (*window*, *slot_nr*)

stacks_with (*other*)

class `spockbot.mdata.windows.SlotCursor` (*id=-1*, *damage=0*, *amount=0*, *enchants=None*)

Bases: `spockbot.mdata.windows.Slot`

class `spockbot.mdata.windows.Window` (*window_id*, *title*, *slot_count*, *inv_type=None*, *persistent_slots=None*, *eid=None*)

Bases: object

Base class for all inventory types.

hotbar_slots

inv_data = {}

inv_type = None

inventory_slots

name = None

persistent_slots

window_slots

All slots except inventory and hotbar. Useful for searching.

`spockbot.mcdata.windows.make_slot_check` (*wanted*)

Creates and returns a function that takes a slot and checks if it matches the wanted item.

Parameters `wanted` – function(Slot) or Slot or itemID or (itemID, metadata)

Module contents

`spockbot.mcdata.get_item_or_block` (*find, meta=0, init=True*)

spockbot.mcp package

Submodules

spockbot.mcp.bbuff module

class `spockbot.mcp.bbuff.BoundBuffer` (*data=b''*)

Bases: `object`

append (*data*)

buff = `b''`

cursor = `0`

flush ()

read (*length*)

recv (*length*)

revert ()

save ()

tell ()

write (*data*)

exception `spockbot.mcp.bbuff.BufferUnderflowException`

Bases: `Exception`

spockbot.mcp.datautils module

`spockbot.mcp.datautils.byte_to_hex` (*byte_str*)

`spockbot.mcp.datautils.pack` (*data_type, data*)

`spockbot.mcp.datautils.pack_fixed_point` (*mc_type, val*)

`spockbot.mcp.datautils.pack_metadata` (*data*)

`spockbot.mcp.datautils.pack_position` (*position*)

`spockbot.mcp.datautils.pack_slot` (*slot*)

`spockbot.mcp.datautils.pack_varint` (*val*)

`spockbot.mcp.datautils.pack_varlong` (*val*)

`spockbot.mcp.datautils.unpack` (*data_type, bbuff*)

`spockbot.mcp.datautils.unpack_fixed_point` (*mc_type, bbuff*)

`spockbot.mcp.datautils.unpack_metadata` (*bbuff*)

`spockbot.mcp.datautils.unpack_position` (*bbuff*)

```
spockbot.mcp.datautils.unpack_slot (bbuff)
spockbot.mcp.datautils.unpack_varint (bbuff)
spockbot.mcp.datautils.unpack_varlong (bbuff)
```

spockbot.mcp.mcdata module

spockbot.mcp.mcpacket module

```
class spockbot.mcp.mcpacket.Packet (ident=[0, 1, 0], data=None)
    Bases: object

    clone ()

    decode (bbuff, proto_comp_state)

    encode (proto_comp_state, proto_comp_threshold, comp_level=6)

    new_ident (ident)
exception spockbot.mcp.mcpacket.PacketDecodeFailure (packet, pbuff, underflow=False)
    Bases: Exception
```

spockbot.mcp.mcpacket_extensions module

spockbot.mcp.nbt module Handle the NBT (Named Binary Tag) data format

```
exception spockbot.mcp.nbt.MalformedFileError
    Bases: Exception

    Exception raised on parse error.

class spockbot.mcp.nbt.Tag (value=None, name=None)
    Bases: object

    Tag, a variable with an intrinsic name.

    __repr__ ()
        Return a string (ascii formatted for Python 2, unicode for Python 3) describing the class, name and id for debugging purposes.

    __str__ ()
        Return a string (ascii formatted for Python 2, unicode for Python 3) with the result in human readable format. Unlike valustr(), the result is recursive for iterators till at least one level deep.

    __unicode__ ()
        Return a unicode string with the result in human readable format. Unlike valustr(), the result is recursive for iterators till at least one level deep.

    id = None

    pretty_tree (indent=0)
        Return formatted Unicode string of self, where iterable items are recursively listed in detail.

    tag_info ()
        Return Unicode string with class, name and unnested value.

    valustr ()
        Return Unicode string of unnested value. For iterators, this returns a summary.
```



```

class spockbot.mcp.nbt.TagByte (value=None, name=None, buffer=None)
    Bases: spockbot.mcp.nbt._TagNumeric

    Represent a single tag storing 1 byte.

    fmt = <Struct object>

    id = 1

class spockbot.mcp.nbt.TagByteArray (name=None, buffer=None)
    Bases: spockbot.mcp.nbt.Tag, collections.abc.MutableSequence

    TagByteArray, comparable to a collections.UserList with an intrinsic name whose values must be bytes

    id = 7

    insert (key, value)

    valustr ()

class spockbot.mcp.nbt.TagCompound (buffer=None)
    Bases: spockbot.mcp.nbt.Tag, collections.abc.MutableMapping

    TagCompound, comparable to a collections.OrderedDict with an intrinsic name

    id = 10

    iteritems ()

    keys ()

    pretty_tree (indent=0)

    valustr ()

class spockbot.mcp.nbt.TagDouble (value=None, name=None, buffer=None)
    Bases: spockbot.mcp.nbt._TagNumeric

    Represent a single tag storing 1 IEEE-754 double precision floating point number.

    fmt = <Struct object>

    id = 6

class spockbot.mcp.nbt.TagFloat (value=None, name=None, buffer=None)
    Bases: spockbot.mcp.nbt._TagNumeric

    Represent a single tag storing 1 IEEE-754 floating point number.

    fmt = <Struct object>

    id = 5

class spockbot.mcp.nbt.TagInt (value=None, name=None, buffer=None)
    Bases: spockbot.mcp.nbt._TagNumeric

    Represent a single tag storing 1 int.

    fmt = <Struct object>

    id = 3

class spockbot.mcp.nbt.TagIntArray (name=None, buffer=None)
    Bases: spockbot.mcp.nbt.Tag, collections.abc.MutableSequence

    TagIntArray, comparable to a collections.UserList with an intrinsic name whose values must be integers

    id = 11

```

insert (*key*, *value*)

update_fmt (*length*)

Adjust struct format description to length given

valuestr ()

class `spockbot.mcp.nbt.TagList` (*type=None*, *value=None*, *name=None*, *buffer=None*)

Bases: `spockbot.mcp.nbt.Tag`, `collections.abc.MutableSequence`

TagList, comparable to a `collections.UserList` with an intrinsic name

id = 9

insert (*key*, *value*)

pretty_tree (*indent=0*)

valuestr ()

class `spockbot.mcp.nbt.TagLong` (*value=None*, *name=None*, *buffer=None*)

Bases: `spockbot.mcp.nbt._TagNumeric`

Represent a single tag storing 1 long.

fmt = <Struct object>

id = 4

class `spockbot.mcp.nbt.TagShort` (*value=None*, *name=None*, *buffer=None*)

Bases: `spockbot.mcp.nbt._TagNumeric`

Represent a single tag storing 1 short.

fmt = <Struct object>

id = 2

class `spockbot.mcp.nbt.TagString` (*value=None*, *name=None*, *buffer=None*)

Bases: `spockbot.mcp.nbt.Tag`, `collections.abc.Sequence`

TagString, comparable to a `collections.UserString` with an intrinsic name

id = 8

spockbot.mcp.yggdrasil module

class `spockbot.mcp.yggdrasil.YggdrasilCore` (*username=''*, *password=''*, *client_token=''*, *access_token=''*)

Bases: `object`

authenticate ()

Generate an access token using an username and password. Any existing client token is invalidated if not provided.

Returns Response or error dict

Return type dict

invalidate ()

Invalidate access tokens with a client/access token pair

Returns Empty or error dict

Return type dict

login ()

logout()

refresh()

Generate an access token with a client/access token pair. Used access token is invalidated.

Returns Response or error dict

Return type dict

signout()

Invalidate access tokens with a username and password.

Returns Empty or error dict

Return type dict

validate()

Check if an access token is valid

Returns Empty or error dict

Return type dict

ygg_url = 'https://authserver.mojang.com'

ygg_version = 1

Module contents

spockbot.plugins package

Subpackages

spockbot.plugins.core package

Submodules

spockbot.plugins.core.auth module Provides authorization functions for Mojang's login and session servers

class spockbot.plugins.core.auth.**AuthCore**(*event, online_mode, auth_timeout*)

Bases: object

auth_token

client_token

get_shared_secret()

get_username()

password

send_session_auth(*pubkey_raw, server_id_raw*)

set_auth_token(*auth_token*)

set_client_token(*client_token*)

set_password(*password*)

set_username(*username*)

```
    shared_secret
    start_session()
    username

class spockbot.plugins.core.auth.AuthPlugin(ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    defaults = {'auth_timeout': 3, 'sess_quit': True, 'auth_quit': True, 'online_mode': True}
    events = {'auth_login_error': 'handle_auth_error', 'auth_session_error': 'handle_session_error'}
    handle_auth_error(name, data)
    handle_session_error(name, data)
    pl_announce = ('Auth',)
    requires = 'Event'

spockbot.plugins.core.auth.java_hex_digest(digest)
```

spockbot.plugins.core.event module Provides the core event loop

```
class spockbot.plugins.core.event.EventPlugin(ploader, settings)
    Bases: object

    emit(event, data=None)
    event_loop(once=False)
    kill(*args)
    pl_announce = ('Event',)
    reg_event_handler(event, handler)
    run_continuous()
    run_once()
    unreg_event_handler(event, handler)
```

spockbot.plugins.core.net module Provides an asynchronous, crypto and compression aware socket for connecting to servers and processing incoming packet data. Coordinates with the Timers plugin to honor wall-clock timers

```
class spockbot.plugins.core.net.AESCipher(shared_secret)
    Bases: object

    decrypt(data)
    encrypt(data)

class spockbot.plugins.core.net.NetCore(sock, event, select)
    Bases: object

    connect(host='localhost', port=25565)
    disable_crypto()
    enable_crypto(secret_key)
    push(packet)
    push_packet(ident, data)
```

```

    read_packet (data=b'')
    reset (sock)
    set_comp_state (threshold)
    set_proto_state (state)
class spockbot.plugins.core.net.NetPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase
    check_quit ()
    defaults = {'sock_quit': True, 'bufsize': 4096}
    events = {'PLAY<Disconnect': 'handle_disconnect', 'LOGIN<Set Compression': 'handle_comp', 'SOCKET_HUP': 'ha
    handle_comp (name, packet)
        Handle Set Compression packets
    handle_disconnect (name, packet)
    handle_err (name, error)
        Socket Error has occurred
    handle_handshake (name, packet)
        Change to whatever the next state is going to be
    handle_hup (name, data)
        Socket has hung up
    handle_kill (name, data)
        Try to shutdown the socket politely
    handle_login_disconnect (name, packet)
    handle_login_success (name, packet)
        Change to Play state
    handle_recv (name, fileno)
        Socket is ready to recieve data
    handle_select_err (name, fileno)
    handle_send (name, fileno)
        Socket is ready to send data and send buffer has data to send
    pl_announce = ('Net',)
    requires = ('Event', 'Select', 'Timers')
    reset_sock ()
    tick (name, data)

spockbot.plugins.core.settings module
class spockbot.plugins.core.settings.PloaderFetch (plugins, plugin_settings)
    Bases: object
    get_plugin_settings (plugin)
    get_plugins ()
class spockbot.plugins.core.settings.SettingsPlugin (ploader, kwargs)
    Bases: object
    pl_announce = ('PloaderFetch',)

```

spockbot.plugins.core.taskmanager module

```
class spockbot.plugins.core.taskmanager.TaskManager (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    pl_announce = ('TaskManager',)

    requires = 'Event'

    run_task (task, parent=None, name=None)
```

spockbot.plugins.core.ticker module Registers timers to provide the necessary tick rates expected by MC servers

```
class spockbot.plugins.core.ticker.TickerPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    client_tick()

    events = {'PLAY<Join Game': 'start_tickers'}

    requires = ('Event', 'Timers')

    start_tickers (_, __)
```

spockbot.plugins.core.timer module

Module contents

spockbot.plugins.helpers package

Submodules

spockbot.plugins.helpers.channels module Provides interface for Plugin Channels

```
class spockbot.plugins.helpers.channels.ChannelsCore (net)
    Bases: object

    decode (structure, data)

    encode (structure, data)

    send (channel, data)
        Send a plugin channel message

class spockbot.plugins.helpers.channels.ChannelsPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    events = {'PLAY<Plugin Message': 'handle_plugin_message'}

    handle_plugin_message (name, packet)

    pl_announce = ('Channels',)

    requires = ('Event', 'Net')
```

spockbot.plugins.helpers.chat module Processes chat into easy to use events

class `spockbot.plugins.helpers.chat.ChatCore` (*net*)

Bases: `object`

chat (*message*)

whisper (*player, message*)

exception `spockbot.plugins.helpers.chat.ChatParseError`

Bases: `Exception`

class `spockbot.plugins.helpers.chat.ChatPlugin` (*ploader, settings*)

Bases: `spockbot.plugins.base.PluginBase`

Emits chat events with position, raw, text, type, message, name, uuid.

position: Always one of `spockbot.mcddata.constants`'s `CHAT_POS_CHAT`, `CHAT_POS_SYSTEM_MESSAGE`, `CHAT_POS_ABOVE_HOTBAR`.

raw: Always the JSON dict as received from the server.

text: The text (without formatting) of the chat message as the vanilla client would display it. Needs `en_US.lang` to be present in the active directory, otherwise some but not all messages are translated properly.

type: None or one of `achievement`, `admin`, `announcement`, `emote`, `incoming`, `outgoing`, `text`, which are the last part of the corresponding vanilla translation IDs.

If type is not None, message, name, uuid are set and an additional `chat_<type>` event is emitted. Otherwise, message, name, uuid are all None.

message: The message as it was typed by the sender.

name: The name of the sender.

uuid: The UUID of the sender, with dashes.

events = {'PLAY<Chat Message>': 'handle_chat'}

handle_chat (*evt, packet*)

load_translations ()

pl_announce = ('Chat',)

render_chat (*chat_data*)

Render the text as in the vanilla client. On a vanilla server, this uses the translations dict.

requires = ('Event', 'Net')

`spockbot.plugins.helpers.chat.parse_with_1_extra` (*json_data*)

spockbot.plugins.helpers.clientinfo module ClientInfo is a central plugin for recording data about the client, e.g. Health, position, and some auxillary information like the player list. Plugins subscribing to ClientInfo's events don't have to independently track this information on their own.

class `spockbot.plugins.helpers.clientinfo.Abilities`

Bases: `spockbot.mcddata.utils.Info`

class `spockbot.plugins.helpers.clientinfo.ClientInfo`

Bases: `object`

eid

int

Entity ID of the player

name
str
Player's Username

uuid
str
Player's UUID

abilities
Abilities
Player's current movement state and speed

game_info
GameInfo
Information about the current world/server

spawn_position
Position
Players initial position

health
PlayerHealth
Player's health, food and saturation

position
PlayerPosition
Player's current position

player_list
dict
List of all players in the server

eye_pos
PlayerPosition
Player's eye position

eye_pos

reset ()
Resets the information in ClientInfo

class `spockbot.plugins.helpers.clientinfo.ClientInfoPlugin` (*ploader, settings*)
Bases: `spockbot.plugins.base.PluginBase`

events = {'PLAY<Server Difficulty': 'handle_server_difficulty', 'LOGIN<Login Success': 'handle_login_success', 'PLA

handle_attach_entity (*name, packet*)

handle_disconnect (*name, data*)

handle_game_state (*name, packet*)

handle_join_game (*name, packet*)

handle_login_success (*name, packet*)

handle_player_abilities (*name, packet*)

handle_player_list (*name, packet*)


```

    handle_position_update (name, packet)
    handle_server_difficulty (name, packet)
    handle_spawn_position (name, packet)
    handle_update_health (name, packet)
    pl_announce = ('ClientInfo',)
    requires = 'Event'

class spockbot.plugins.helpers.clientinfo.GameInfo
    Bases: spockbot.mcddata.utils.Info

class spockbot.plugins.helpers.clientinfo.PlayerHealth
    Bases: spockbot.mcddata.utils.Info

class spockbot.plugins.helpers.clientinfo.PlayerListItem
    Bases: spockbot.mcddata.utils.Info

class spockbot.plugins.helpers.clientinfo.PlayerPosition (*xyz)
    Bases: spockbot.plugins.helpers.clientinfo.Position

class spockbot.plugins.helpers.clientinfo.Position (*xyz)
    Bases: spockbot.vector.Vector3, spockbot.mcddata.utils.Info

    Used for things that require encoding position for the protocol, but also require higher level vector functions.

    get_dict ()

```

spockbot.plugins.helpers.craft module Craft items.

```

class spockbot.plugins.helpers.craft.CraftPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    craft (item=None, meta=None, amount=1, recipe=None, parent=None)
        Starts a craft_task. Either item or recipe has to be given.

        Returns The recipe used for crafting.

        Return type Optional[Recipe]

    craft_task (recipe, amount=1)
        A task that crafts amount items with recipe.

    pl_announce = ('Craft',)

    requires = ('Inventory', 'TaskManager')

```

spockbot.plugins.helpers.entities module An entity tracker

```

class spockbot.plugins.helpers.entities.EntitiesCore
    Bases: object

class spockbot.plugins.helpers.entities.EntitiesPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    events = {'PLAY<Spawn Global Entity': 'handle_spawn_global_entity', 'PLAY<Spawn Experience Orb': 'handle_spawn_experience_orb'}

    handle_destroy_entities (event, packet)

    handle_join_game (event, packet)

    handle_relative_move (event, packet)

```

```

    handle_set_dict (event, packet)
    handle_spawn_experience_orb (event, packet)
    handle_spawn_global_entity (event, packet)
    handle_spawn_mob (event, packet)
    handle_spawn_object (event, packet)
    handle_spawn_painting (event, packet)
    handle_spawn_player (event, packet)
    handle_unhandled (event, packet)
    handle_velocity (event, packet)
    pl_announce = ('Entities',)
    requires = 'Event'

class spockbot.plugins.helpers.entities.ExpEntity
    Bases: spockbot.plugins.helpers.entities.MCEntity
    count = 0
    x = 0
    y = 0
    z = 0

class spockbot.plugins.helpers.entities.GlobalEntity
    Bases: spockbot.plugins.helpers.entities.MCEntity
    global_type = 0
    x = 0
    y = 0
    z = 0

class spockbot.plugins.helpers.entities.MCEntity
    Bases: spockbot.mcdata.utils.Info
    eid = 0
    metadata = None
    nbt = None
    status = 0

class spockbot.plugins.helpers.entities.MobEntity
    Bases: spockbot.plugins.helpers.entities.MovementEntity
    head_pitch = 0
    head_yaw = 0
    metadata = None
    mob_type = 0
    velocity_x = 0
    velocity_y = 0

```

```

    velocity_z = 0
class spockbot.plugins.helpers.entities.MovementEntity
    Bases: spockbot.plugins.helpers.entities.MCEntity
    on_ground = True
    pitch = 0
    x = 0
    y = 0
    yaw = 0
    z = 0
class spockbot.plugins.helpers.entities.ObjectEntity
    Bases: spockbot.plugins.helpers.entities.MovementEntity
    obj_data = 0
    obj_type = 0
    speed_x = 0
    speed_y = 0
    speed_z = 0
class spockbot.plugins.helpers.entities.PaintingEntity
    Bases: spockbot.plugins.helpers.entities.MCEntity
    direction = 0
    location = {'y': 0, 'z': 0, 'x': 0}
    title = ''
class spockbot.plugins.helpers.entities.PlayerEntity
    Bases: spockbot.plugins.helpers.entities.MovementEntity
    current_item = 0
    metadata = None
    uuid = 0

```

spockbot.plugins.helpers.interact module Interact with the world: - swing the arm, sneak, sprint, jump with a horse, leave the bed - look around - dig/place/use blocks - use the held (active) item - use/attack entities - steer vehicles - place and write signs - edit and sign books

By default, the client sends swing and look packets like the vanilla client. This can be disabled by setting the `auto_swing` and `auto_look` flags.

```

class spockbot.plugins.helpers.interact.InteractPlugin(ploader, settings)
    Bases: spockbot.plugins.base.PluginBase
    activate_item()
        Use (hold right-click) the item in the active slot. Examples: pull the bow, start eating once, throw an egg.
    attack_entity(entity)
    cancel_digging()
    click_block(pos, look_at_block=True, swing=True, **kwargs)
        Click on a block. Examples: push button, open window, make redstone ore glow

```

Parameters

- **face** (*int*) – side of the block on which the block is placed on
- **cursor_pos** (*Vector3*) – where to click inside the block, each dimension 0-15

deactivate_item ()

Stop using (release right-click) the item in the active slot. Examples: shoot the bow, stop eating.

dig_block (*pos*)

Not cancelable.

edit_book (*pages*)

Set the pages of current book in hand

finish_digging ()

jump_horse (*jump_boost=100*)

jump_vehicle ()

leave_bed ()

look (*yaw=0.0, pitch=0.0, radians=False*)

look_at (*pos*)

look_at_rel (*delta*)

look_rel (*d_yaw=0.0, d_pitch=0.0, radians=False*)

mount_vehicle (*entity*)

open_inventory ()

pl_announce = ('Interact',)

place_block (*pos, sneak=True, **kwargs*)

Place a block next to *pos*. If the block at *pos* is air, place at *pos*.

place_sign (*pos, lines=[], **place_block_kwargs*)

Place a sign block and write on it.

requires = ('ClientInfo', 'Event', 'Inventory', 'Net', 'Channels')

sign_book (*author, title*)

Sign current book in hand

sneak (*sneak=True*)

sprint (*sprint=True*)

start_digging (*pos, face=1*)

steer_vehicle (*left=0.0, forward=0.0, jump=False, unmount=False*)

swing_arm ()

unmount_vehicle ()

unsneak ()

unsprint ()

use_bucket (*pos*)

Using buckets is different from placing blocks. See “Special note on using buckets” in http://wiki.vg/Protocol#Player_Block_Placement

use_entity (*entity*, *cursor_pos=None*, *action=0*)

Uses (right-click) an entity to open its window. Setting *cursor_pos* sets *action* to “interact at”.

write_book (*text*, *author=''*, *title=''*, *sign=False*)

Write text to the current book in hand, optionally sign the book

spockbot.plugins.helpers.inventory module The Inventory plugin keeps track of the inventory and provides simple inventory analysis and manipulation.

class `spockbot.plugins.helpers.inventory.InventoryCore` (*net_plugin*, *send_click*)

Bases: object

Handles operations with the player inventory.

active_slot

click_slot (*slot*, *right=False*)

Left-click or right-click the slot.

Parameters *slot* (*Slot*) – The clicked slot. Can be *Slot* instance or integer. Set to `inventory.cursor_slot` for clicking outside the window.

close_window ()

creative_set_slot (*slot_nr=None*, *slot_dict=None*, *slot=None*)

drop_slot (*slot=None*, *drop_stack=False*)

Drop one or all items of the slot.

Does not wait for confirmation from the server. If you want that, use a *Task* and `yield inventory.async.drop_slot()` instead.

If *slot* is *None*, drops the `cursor_slot` or, if that’s empty, the currently held item (`active_slot`).

Parameters *slot* (*Optional[Slot]*) – The dropped slot. Can be *None*, integer, or *Slot* instance.

Returns The action ID of the click

Return type *int*

find_slot (*wanted*, *slots=None*)

Searches the given slots or, if not given, active hotbar slot, hotbar, inventory, open window in this order.

Parameters *wanted* – function(*Slot*) or *Slot* or *itemID* or (*itemID*, *metadata*)

Returns The first slot containing the item or *None* if not found.

Return type *Optional[Slot]*

find_slots (*wanted*, *slots=None*)

Yields all slots containing the item. Searches the given slots or, if not given, active hotbar slot, hotbar, inventory, open window in this order.

Parameters *wanted* – function(*Slot*) or *Slot* or *itemID* or (*itemID*, *metadata*)

inv_slots_preferred

List of all available inventory slots in the preferred search order. Does not include the additional slots from the open window.

- 1.active slot
- 2.remainder of the hotbar
- 3.remainder of the persistent inventory

select_active_slot (*slot_or_hotbar_index*)

total_stored (*wanted, slots=None*)

Calculates the total number of items of that type in the current window or given slot range.

Parameters **wanted** – function(Slot) or Slot or itemID or (itemID, metadata)

class `spockbot.plugins.helpers.inventory.InventoryPlugin` (*ploader, settings*)

Bases: `spockbot.plugins.base.PluginBase`

emit_open_window (*_)

emit_set_slot (*slot*)

events = {'PLAY<Open Window': 'handle_open_window', 'PLAY<Held Item Change': 'handle_held_item_change', 'P

handle_close_window (*event, packet*)

handle_confirm_transaction (*event, packet*)

handle_held_item_change (*event, packet*)

handle_open_window (*event, packet*)

handle_set_slot (*event, packet*)

handle_window_items (*event, packet*)

handle_window_prop (*event, packet*)

pl_announce = ('Inventory',)

requires = ('Event', 'Net', 'Timers')

send_click (*click*)

Sends a click to the server if the previous click has been confirmed.

Parameters **click** (`BaseClick`) – The click to send.

Returns the click's action ID if the click could be sent, None if the previous click has not been received and confirmed yet.

set_slot (*window_id, slot_nr, slot_data*)

spockbot.plugins.helpers.keepalive module

spockbot.plugins.helpers.movement module MovementPlugin provides a centralized plugin for controlling client movement so the client doesn't try to pull itself in a dozen directions.

class `spockbot.plugins.helpers.movement.MovementCore` (*plug*)

Bases: object

current_path

current_target

final_target

is_moving

stop ()

class `spockbot.plugins.helpers.movement.MovementPlugin` (*ploader, settings*)

Bases: `spockbot.plugins.base.PluginBase`

follow_path (_, __)

```

new_path (*xyz)
path_cb (result)
pl_announce = ('Movement',)
requires = ('ClientInfo', 'Event', 'Net', 'Pathfinding', 'Physics')

```

spockbot.plugins.helpers.pathfinding module Very rough asynchronous pathfinding plugin Implements the Lazy Theta* pathfinding algorithm

```

class spockbot.plugins.helpers.pathfinding.Path (start_node, end_node)
    Bases: object

    calc_f_val (node)

class spockbot.plugins.helpers.pathfinding.PathNode (*xyz)
    Bases: spockbot.vector.Vector3

    set (parent=None, is_fall=False, is_jump=False)

class spockbot.plugins.helpers.pathfinding.PathfindingCore (start_path)
    Bases: object

class spockbot.plugins.helpers.pathfinding.PathfindingPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    build_list_from_node (node)

    check_for_bbox (pos)

    check_node (node, offset, node_list, walk_fall=True, jump=True)

    do_job (_=None, __=None)

    find_valid_nodes (node)

    get_block (pos)

    pathfind (path)

    pl_announce = ('Pathfinding',)

    raycast_bbox (start, end)

    requires = ('Event', 'World', 'Physics', 'ClientInfo', 'Timers')

    start_path (pos, target, scb, fcb=None)

```

spockbot.plugins.helpers.physics module A Physics module built from clean-rooming the Notchian Minecraft client

Collision detection and resolution is done by a Separating Axis Theorem implementation for concave shapes decomposed into Axis-Aligned Bounding Boxes. This isn't totally equivalent to vanilla behavior, but it's faster and Close Enough^TM

AKA this file does Minecraft physics

```

class spockbot.plugins.helpers.physics.PhysicsCore (pos, vec, abilities)
    Bases: object

    jump ()

    move_angle (angle, radians=False)

    move_target (vector)

```

```

    move_vector (vector)

    sprint ()

    walk ()

class spockbot.plugins.helpers.physics.PhysicsPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    apply_accel ()

    apply_drag ()

    apply_vector (mtv)

    client_tick (name, data)

    events = {'client_unmount': 'resume_physics', 'client_position_update': 'skip_physics', 'physics_tick': 'physics_tick', '

    get_block_slip ()

    get_mtv ()

    physics_tick (_, __)

    pl_announce = ('Physics',)

    requires = ('Event', 'ClientInfo', 'Net', 'World')

    resume_physics (_=None, __=None)

    skip_physics (_=None, __=None)

    suspend_physics (_=None, __=None)

```

spockbot.plugins.helpers.respawn module

spockbot.plugins.helpers.start module This plugin creates a convenient start() method and attaches it directly to the client. More complex bots will likely want to create their own initialization plugin, so StartPlugin stays out of the way unless you call the start() method. However, the start() method is very convenient for demos and tutorials, and illustrates the basic steps for initializing a bot.

```

class spockbot.plugins.helpers.start.StartPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase

    defaults = {'username': 'Bot', 'password': None, 'port': 25565, 'host': 'localhost'}

    events = {'event_start': 'start_session_and_connect'}

    requires = ('Auth', 'Event', 'Net')

    start (host=None, port=None)

    start_session_and_connect (_, __)

```

spockbot.plugins.helpers.world module Provides a very raw (but very fast) world map for use by plugins. Plugins interested in a more comprehensive world map view can use mcp.mapdata to interpret blocks and their metadata more comprehensively. Planned to provide light level interpretation based on sky light and time of day

```

class spockbot.plugins.helpers.world.WorldData (dimension=0)
    Bases: spockbot.plugins.tools.smpmap.Dimension

    new_dimension (dimension)

```



```

    reset ()
    update_time (data)
class spockbot.plugins.helpers.world.WorldPlugin (ploader, settings)
    Bases: spockbot.plugins.base.PluginBase
    events = {'PLAY<Time Update': 'handle_time_update', 'PLAY<Update Sign': 'handle_update_sign', 'net_disconnect':
    handle_block_change (name, packet)
        Block Change - Update a single block
    handle_chunk_data (name, packet)
        Chunk Data - Update World state
    handle_disconnect (name, data)
    handle_map_chunk_bulk (name, packet)
        Map Chunk Bulk - Update World state
    handle_multi_block_change (name, packet)
        Multi Block Change - Update multiple blocks
    handle_new_dimension (name, packet)
        Join Game/Respawn - New Dimension
    handle_time_update (name, packet)
        Time Update - Update World Time
    handle_update_block_entity (event, packet)
    handle_update_sign (event, packet)
    pl_announce = ('World',)
    requires = 'Event'

```

Module contents

spockbot.plugins.tools package

Submodules

spockbot.plugins.tools.collision module

```

class spockbot.plugins.tools.collision.MTVTest (world, bbox)
    Bases: object
    block_collision (pos)
    check_collision (pos, vector)
spockbot.plugins.tools.collision.center_position (pos, bbox)
spockbot.plugins.tools.collision.check_axis (axis, min_a, max_a, min_b, max_b)
spockbot.plugins.tools.collision.gen_block_set (block_pos, xr=(-1, 2), yr=(0, 3), zr=(-1,
2))
spockbot.plugins.tools.collision.uncenter_position (pos, bbox)

```

spockbot.plugins.tools.event module Used for unregistering event handlers

spockbot.plugins.tools.inventory_async module Asynchronous task wrappers for inventory

class `spockbot.plugins.tools.inventory_async.InventoryAsync` (*inventory*)

Bases: `object`

click_slot (*slot*, *right=False*)

click_slots (**slots*)

creative_set_slot (*slot_nr=None*, *slot_dict=None*, *slot=None*)

drop_slot (*slot=None*, *drop_stack=False*)

hold_item (*wanted*)

move_to_inventory (**slots*)

move_to_window (**slots*)

store_or_drop ()

Stores the cursor item or drops it if the inventory is full. Tip: look directly up or down before calling this, so you can pick up the dropped item when the inventory frees up again.

Returns The slot used to store it, or None if dropped.

Return type *Slot*

swap_slots (*a*, *b*)

transfer_slots (*source_slots*, *target_slots*)

`spockbot.plugins.tools.inventory_async.unpack_slots_list` (*slots*)

spockbot.plugins.tools.smpmap module Used for storing map data

Chunks are packed in X, Z, Y order The array walks down X, every 16 elements you enter a new Z-level ex. [0] - [15] are X = 0-15, Z = 0, Y = 0 [16] - [31] are X = 0-15, Z = 1, Y = 0 and so on

Every 256 elements you enter a new Y-level ex. [0]-[255] are X = 0-15, Z = 0-15, Y = 0 [256]-[511] are X = 0-15, Z = 0-15, Y = 1 and so on

class `spockbot.plugins.tools.smpmap.BannerData` (*nbt*)

Bases: `spockbot.plugins.tools.smpmap.BlockEntityData`

class `spockbot.plugins.tools.smpmap.BeaconData` (*nbt*)

Bases: `spockbot.plugins.tools.smpmap.BlockEntityData`

class `spockbot.plugins.tools.smpmap.BiomeData`

Bases: `spockbot.plugins.tools.smpmap.ChunkData`

A 16x16 array stored in each ChunkColumn.

data = None

get (*x*, *z*)

length = 256

set (*x*, *z*, *d*)

class `spockbot.plugins.tools.smpmap.BlockEntityData` (*nbt*)

Bases: `object`

class `spockbot.plugins.tools.smpmap.Chunk`

Bases: `object`

```

class spockbot.plugins.tools.smpmap.ChunkColumn
    Bases: object

    unpack (buff, mask, skylight=True, continuous=True)

class spockbot.plugins.tools.smpmap.ChunkData
    Bases: object

    data = None

    fill ()

    get (x, y, z)

    length = 4096

    pack ()

    set (x, y, z, data)

    ty = 'B'

    unpack (buff)

class spockbot.plugins.tools.smpmap.ChunkDataNibble
    Bases: spockbot.plugins.tools.smpmap.ChunkData

    A 16x16x8 array for storing metadata, light or add. Each array element contains two 4-bit elements.

    get (x, y, z)

    length = 2048

    set (x, y, z, data)

class spockbot.plugins.tools.smpmap.ChunkDataShort
    Bases: spockbot.plugins.tools.smpmap.ChunkData

    A 16x16x16 array for storing block IDs/Metadata.

    length = 8192

    ty = 'H'

class spockbot.plugins.tools.smpmap.CommandBlockData (nbt)
    Bases: spockbot.plugins.tools.smpmap.BlockEntityData

class spockbot.plugins.tools.smpmap.Dimension (dimension)
    Bases: object

    A bunch of ChunkColumns.

    get_biome (x, z)

    get_block (pos_or_x, y=None, z=None)

    get_block_entity_data (pos_or_x, y=None, z=None)
        Access block entity data.

        Returns BlockEntityData subclass instance or None if no block entity data is stored for that
        location.

    get_light (pos_or_x, y=None, z=None)

    set_biome (x, z, data)

    set_block (pos_or_x, y=None, z=None, block_id=None, meta=None, data=None)

```

set_block_entity_data (*pos_or_x*, *y=None*, *z=None*, *data=None*)

Update block entity data.

Returns Old data if block entity data was already stored for that location, None otherwise.

set_light (*pos_or_x*, *y=None*, *z=None*, *light_block=None*, *light_sky=None*)

unpack_bulk (*data*)

unpack_column (*data*)

class spockbot.plugins.tools.smpmap.**FlowerPotData** (*nbt*)

Bases: *spockbot.plugins.tools.smpmap.BlockEntityData*

class spockbot.plugins.tools.smpmap.**HeadData** (*nbt*)

Bases: *spockbot.plugins.tools.smpmap.BlockEntityData*

class spockbot.plugins.tools.smpmap.**SignData** (*line_data*)

Bases: *spockbot.plugins.tools.smpmap.BlockEntityData*

class spockbot.plugins.tools.smpmap.**SpawnerData** (*nbt*)

Bases: *spockbot.plugins.tools.smpmap.BlockEntityData*

spockbot.plugins.tools.smpmap.**mapshort2id** (*data*)

spockbot.plugins.tools.task module

class spockbot.plugins.tools.task.**Task** (*task*, *parent=None*, *name=None*)

Bases: object

continue_with (*func*)

on_error (*exception*)

on_event (*event*, *data*)

on_success (*data*)

parse_response (*response*)

register (*response*)

run (*task_manager*)

tasktrace

List of all parent tasks up to this one.

Returns List[Task]

class spockbot.plugins.tools.task.**TaskCallback** (*cb=None*, *eb=None*)

Bases: object

on_error (*error*)

on_success (*data*)

exception spockbot.plugins.tools.task.**TaskFailed** (*message*, **args*)

Bases: Exception

Raising this exception in any task stops it and signalizes the parent task that the task was aborted due to an error.

message

str

Description of the failure

tasktrace*List[Task]*

List of all failed tasks since raising this error.

prev_error*TaskFailed*

The previous error, if any. Provide via `with_error()`.

__str__()

Newline-separated text with all failed tasks and all previous errors.

full_tasktrace

List of all failed tasks caused by this and all previous errors.

Returns *List[Task]*

with_error (*prev_error*)

Sets the previous error and returns self.

When re-throwing a `TaskFailed`, you can provide a new, more high level failure description and pass along the previously failed tasks to still be able to reconstruct the full history of failed tasks.

Examples

Re-throw a `TaskFailed` with a new, more high level description.

```
>>> try:
...     raise TaskFailed('Low level', {'some': 1}, 'args')
... except TaskFailed as prev_err:
...     raise TaskFailed('High level').with_error(prev_err)
```

Returns *TaskFailed*

`spockbot.plugins.tools.task.accept` (*evt, data*)

`spockbot.plugins.tools.task.check_key` (*key, value*)

Generates a check function for a certain key-value pair.

Creates and returns a function that takes two arguments (*event, data*) and checks `data[key]` and *value* for equality.

This is supposed to be used as a check function generator for the `yield` statements in tasks.

Example

Wait for the next `player_join` event that has its name set to Bob, i.e. `data = {'name': 'Bob'}`.

```
>>> def my_task():
...     yield 'player_join', check_key('name', 'Bob')
```

Module contents**Submodules**

spockbot.plugins.base module

class spockbot.plugins.base.**PluginBase** (*ploader, settings*)

Bases: object

A base class for cleaner plugin code.

Extending from PluginBase allows you to declare any requirements, default settings, and event listeners in a declarative way. Define the appropriate attributes on your subclass and enjoy cleaner code.

defaults = {}

events = {}

requires = ()

spockbot.plugins.base.**get_settings** (*defaults, settings*)

spockbot.plugins.base.**pl_announce** (**args*)

spockbot.plugins.base.**pl_event** (**args*)

spockbot.plugins.loader module Provides reasonably not-awful plugin loading

class spockbot.plugins.loader.**PluginLoader** (***kwargs*)

Bases: object

provides (*ident, obj*)

requires (*ident, hard=True, warning=None*)

Module contents

Submodules

spockbot.vector module

class spockbot.vector.**BaseVector** (**values*)

Bases: object

init (**args*)

class spockbot.vector.**CartesianVector** (**values*)

Bases: *spockbot.vector.BaseVector*

ceil ()

dist (*other=None*)

dist_cubic (*other=None*)

Manhattan distance

dist_sq (*other=None*)

For fast length comparison

dot_product (*other*)

floor ()

iadd (*other*)

iceil ()

idiv (*other*)

```
    ifloor()
    imul(other)
    isub(other)
    itruediv(other)
    norm()
    trunc()
    zero()

class spockbot.vector.Vector3(*xyz)
    Bases: spockbot.vector.CartesianVector

    get_dict()
    init(*args)
    set_dict(data)
    x
    y
    yaw_pitch
        Calculate the yaw and pitch of this vector
    z

class spockbot.vector.YawPitch(*args)
    Bases: spockbot.vector.BaseVector

    Store the yaw and pitch (in degrees)

    init(*args)

    pitch
        Pitch in degrees

    rpitch
        Pitch in radians

    ryaw
        Yaw in radians

    unit_vector()
        Generate a unit vector (norm = 1)

    yaw
        Yaw in degrees
```

Module contents

1.5 Contributing

1.5.1 Fork, Clone

Fork the SpockBot repository, then clone your fork locally.

1.5.2 Setup

Configure remote

```
$ cd SpockBot
$ git remote add upstream git://github.com/SpockBotMC/SpockBot
```

Install development dependencies

```
$ pip3 install -r dev-requirements.txt
```

Install extra test interpreters

Installing the extra interpreters allows tox to test against multiple versions of python.

```
$ sudo apt-get install python2.7 python3.3 python3.4
```

1.5.3 Development

Create development branch

Do not work in the `master` branch, create a bug or feature branch and work from there. `master` should always be a mirror of upstream master.

```
$ git checkout -b feature-tacocat
```

Please use a more descriptive branch name than `feature-tacocat`

Hack away at the code

Have fun!

Docstrings

We use Google Style Python Docstrings, an example can be found [here](#)

Test your changes

Running `tox` will test that your changes pass all tests and follows the pep8 standard.

```
$ tox
```

Pull Request

If `tox` comes back with a success you are ready for a pull request. Commit your changes to your feature branch, push them to your fork and create a pull request.

1.5.4 Testing

We highly encourage writing tests, we use the pytest framework and you can run `tox` to test your newly written tests.

1.6 Authors

The following persons have contributed to SpockBot. The list is in the order of first contribution. For details on who contributed what, please refer to our Git repository.

- Nicholas Gamberini <nick@gamberini.email>
- Morgan Creekmore <morgan@morgancreekmore.me>
- gjum <code.gjum@gmail.com>
- Jasper Bok <hello@jasperbok.nl>
- Javier Domingo <javierdo1@gmail.com>
- Joshua Aune <luken@omner.org>

1.7 License

Copyright (C) 2015 The SpockBot Project

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Indices and tables

- `genindex`
- `modindex`

m

- `spockbot.mcdata`, 19
- `spockbot.mcdata.biomes`, 15
- `spockbot.mcdata.blocks`, 15
- `spockbot.mcdata.constants`, 16
- `spockbot.mcdata.items`, 16
- `spockbot.mcdata.materials`, 16
- `spockbot.mcdata.recipes`, 16
- `spockbot.mcdata.utils`, 17
- `spockbot.mcdata.windows`, 17
- `spockbot.mcp`, 23
- `spockbot.mcp.bbuff`, 19
- `spockbot.mcp.datautils`, 19
- `spockbot.mcp.mcpacket`, 20
- `spockbot.mcp.nbt`, 20
- `spockbot.mcp.yggdrasil`, 22

p

- `spockbot.plugins`, 42
- `spockbot.plugins.base`, 42
- `spockbot.plugins.core`, 26
- `spockbot.plugins.core.auth`, 1
- `spockbot.plugins.core.event`, 2
- `spockbot.plugins.core.net`, 2
- `spockbot.plugins.core.settings`, 3
- `spockbot.plugins.core.taskmanager`, 3
- `spockbot.plugins.core.ticker`, 4
- `spockbot.plugins.helpers`, 37
- `spockbot.plugins.helpers.channels`, 4
- `spockbot.plugins.helpers.chat`, 5
- `spockbot.plugins.helpers.clientinfo`, 6
- `spockbot.plugins.helpers.craft`, 8
- `spockbot.plugins.helpers.entities`, 9
- `spockbot.plugins.helpers.interact`, 9
- `spockbot.plugins.helpers.inventory`, 11
- `spockbot.plugins.helpers.movement`, 12
- `spockbot.plugins.helpers.pathfinding`, 13
- `spockbot.plugins.helpers.physics`, 13
- `spockbot.plugins.helpers.start`, 14

- `spockbot.plugins.helpers.world`, 14
- `spockbot.plugins.loader`, 42
- `spockbot.plugins.tools`, 41
- `spockbot.plugins.tools.collision`, 37
- `spockbot.plugins.tools.event`, 37
- `spockbot.plugins.tools.inventory_async`, 38
- `spockbot.plugins.tools.smpmap`, 38
- `spockbot.plugins.tools.task`, 40

s

- `spockbot`, 43

v

- `spockbot.vector`, 42

Symbols

[__getnewargs__\(\)](#) (spockbot.mcdata.recipes.RecipeItem method), 16
[__getstate__\(\)](#) (spockbot.mcdata.recipes.RecipeItem method), 16
[__new__\(\)](#) (spockbot.mcdata.recipes.RecipeItem static method), 16
[__repr__\(\)](#) (spockbot.mcdata.recipes.RecipeItem method), 17
[__repr__\(\)](#) (spockbot.mcp.nbt.Tag method), 20
[__str__\(\)](#) (spockbot.mcp.nbt.Tag method), 20
[__str__\(\)](#) (spockbot.plugins.tools.task.TaskFailed method), 41
[__unicode__\(\)](#) (spockbot.mcp.nbt.Tag method), 20

A

Abilities (class in spockbot.plugins.helpers.clientinfo), 27
 abilities (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 28
[accept\(\)](#) (in module spockbot.plugins.tools.task), 41
[activate_item\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 31
[active_slot](#) (spockbot.plugins.helpers.inventory.InventoryCore attribute), 11, 33
 AESCipher (class in spockbot.plugins.core.net), 24
[amount](#) (spockbot.mcdata.recipes.RecipeItem attribute), 17
[append\(\)](#) (spockbot.mcp.bbuff.BoundsBuffer method), 19
[apply\(\)](#) (spockbot.mcdata.windows.BaseClick method), 17
[apply\(\)](#) (spockbot.mcdata.windows.DropClick method), 18
[apply\(\)](#) (spockbot.mcdata.windows.SingleClick method), 18
[apply_accel\(\)](#) (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
[apply_drag\(\)](#) (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
[apply_vector\(\)](#) (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36

[attack_entity\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 31
[auth_token](#) (spockbot.plugins.core.auth.AuthCore attribute), 2, 23
 AuthCore (class in spockbot.plugins.core.auth), 2, 23
[authenticate\(\)](#) (spockbot.mcp.yggdrasil.YggdrasilCore method), 22
 AuthPlugin (class in spockbot.plugins.core.auth), 24

B

BannerData (class in spockbot.plugins.tools.smpmap), 38
 BaseClick (class in spockbot.mcdata.windows), 17
 BaseVector (class in spockbot.vector), 42
 BeaconData (class in spockbot.plugins.tools.smpmap), 38
 BiomeData (class in spockbot.plugins.tools.smpmap), 38
 Block (class in spockbot.mcdata.blocks), 15
[block_collision\(\)](#) (spockbot.plugins.tools.collision.MTVTest method), 37
[block_ext\(\)](#) (in module spockbot.mcdata.blocks), 16
 BlockEntityData (class in spockbot.plugins.tools.smpmap), 38
 BoundBuffer (class in spockbot.mcp.bbuff), 19
[bounding_box](#) (spockbot.mcdata.blocks.Block attribute), 15
 BoundingBox (class in spockbot.mcdata.utils), 17
[buff](#) (spockbot.mcp.bbuff.BoundsBuffer attribute), 19
 BufferUnderflowException, 19
[build_list_from_node\(\)](#) (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
[byte_to_hex\(\)](#) (in module spockbot.mcp.datautils), 19

C

[cancel_f_val\(\)](#) (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
[cancel_case\(\)](#) (in module spockbot.mcdata.utils), 17
[cancel_digging\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 31

- CartesianVector (class in spockbot.vector), 42
 - ceil() (spockbot.vector.CartesianVector method), 42
 - center_position() (in module spockbot.plugins.tools.collision), 37
 - ChannelsCore (class in spockbot.plugins.helpers.channels), 5, 26
 - ChannelsPlugin (class in spockbot.plugins.helpers.channels), 26
 - chat() (spockbot.plugins.helpers.chat.ChatCore method), 6, 27
 - ChatCore (class in spockbot.plugins.helpers.chat), 6, 27
 - ChatParseError, 27
 - ChatPlugin (class in spockbot.plugins.helpers.chat), 27
 - check_axis() (in module spockbot.plugins.tools.collision), 37
 - check_collision() (spockbot.plugins.tools.collision.MTVTest method), 37
 - check_for_bbox() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
 - check_key() (in module spockbot.plugins.tools.task), 41
 - check_node() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
 - check_quit() (spockbot.plugins.core.net.NetPlugin method), 25
 - Chunk (class in spockbot.plugins.tools.smpmap), 38
 - ChunkColumn (class in spockbot.plugins.tools.smpmap), 38
 - ChunkData (class in spockbot.plugins.tools.smpmap), 39
 - ChunkDataNibble (class in spockbot.plugins.tools.smpmap), 39
 - ChunkDataShort (class in spockbot.plugins.tools.smpmap), 39
 - clean_var() (in module spockbot.mcdata.utils), 17
 - cleanup_if_empty() (spockbot.mcdata.windows.BaseClick method), 17
 - click_block() (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 31
 - click_slot() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33
 - click_slot() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
 - click_slots() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
 - client_tick() (spockbot.plugins.core.ticker.TickerPlugin method), 26
 - client_tick() (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
 - client_token (spockbot.plugins.core.auth.AuthCore attribute), 2, 23
 - ClientInfo (class in spockbot.plugins.helpers.clientinfo), 7, 27
 - ClientInfoPlugin (class in spockbot.plugins.helpers.clientinfo), 28
 - clone() (spockbot.mcp.mcpacket.Packet method), 20
 - close_window() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33
 - CommandBlockData (class in spockbot.plugins.tools.smpmap), 39
 - connect() (spockbot.plugins.core.net.NetCore method), 3, 24
 - continue_with() (spockbot.plugins.tools.task.Task method), 40
 - copy() (spockbot.mcdata.windows.Slot method), 18
 - copy_slot_type() (spockbot.mcdata.windows.BaseClick method), 17
 - count (spockbot.plugins.helpers.entities.ExpEntity attribute), 30
 - craft() (spockbot.plugins.helpers.craft.CraftPlugin method), 8, 29
 - craft_task() (spockbot.plugins.helpers.craft.CraftPlugin method), 8, 29
 - CraftPlugin (class in spockbot.plugins.helpers.craft), 8, 29
 - creative_set_slot() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33
 - creative_set_slot() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
 - current_item (spockbot.plugins.helpers.entities.PlayerEntity attribute), 31
 - current_path (spockbot.plugins.helpers.movement.MovementCore attribute), 12, 34
 - current_target (spockbot.plugins.helpers.movement.MovementCore attribute), 12, 34
 - cursor (spockbot.mcp.bbuff.BoundBuffer attribute), 19
- ## D
- data (spockbot.plugins.tools.smpmap.BiomeData attribute), 38
 - data (spockbot.plugins.tools.smpmap.ChunkData attribute), 39
 - deactivate_item() (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 32
 - decode() (spockbot.mcp.mcpacket.Packet method), 20
 - decode() (spockbot.plugins.helpers.channels.ChannelsCore method), 5, 26
 - decrypt() (spockbot.plugins.core.net.AESCipher method), 24
 - defaults (spockbot.plugins.base.PluginBase attribute), 42
 - defaults (spockbot.plugins.core.auth.AuthPlugin attribute), 24

defaults (spockbot.plugins.core.net.NetPlugin attribute), 25

defaults (spockbot.plugins.helpers.start.StartPlugin attribute), 36

dig_block() (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 32

diggable (spockbot.mcpdata.blocks.Block attribute), 15

Dimension (class in spockbot.plugins.tools.smpmap), 39

direction (spockbot.plugins.helpers.entities.PaintingEntity attribute), 31

disable_crypto() (spockbot.plugins.core.net.NetCore method), 3, 24

display_name (spockbot.mcpdata.blocks.Block attribute), 15

display_name (spockbot.mcpdata.items.Item attribute), 16

dist() (spockbot.vector.CartesianVector method), 42

dist_cubic() (spockbot.vector.CartesianVector method), 42

dist_sq() (spockbot.vector.CartesianVector method), 42

do_job() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35

dot_product() (spockbot.vector.CartesianVector method), 42

drop_slot() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33

drop_slot() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38

DropClick (class in spockbot.mcpdata.windows), 18

drops (spockbot.mcpdata.blocks.Block attribute), 15

EntitiesPlugin (class in spockbot.plugins.helpers.entities), 29

event_loop() (spockbot.plugins.core.event.EventPlugin method), 24

EventPlugin (class in spockbot.plugins.core.event), 24

events (spockbot.plugins.base.PluginBase attribute), 42

events (spockbot.plugins.core.auth.AuthPlugin attribute), 24

events (spockbot.plugins.core.net.NetPlugin attribute), 25

events (spockbot.plugins.core.ticker.TickerPlugin attribute), 26

events (spockbot.plugins.helpers.channels.ChannelsPlugin attribute), 26

events (spockbot.plugins.helpers.chat.ChatPlugin attribute), 27

events (spockbot.plugins.helpers.clientinfo.ClientInfoPlugin attribute), 28

events (spockbot.plugins.helpers.entities.EntitiesPlugin attribute), 29

Events (spockbot.plugins.helpers.inventory.InventoryPlugin attribute), 34

events (spockbot.plugins.helpers.physics.PhysicsPlugin attribute), 36

events (spockbot.plugins.helpers.start.StartPlugin attribute), 36

events (spockbot.plugins.helpers.world.WorldPlugin attribute), 37

ExpEntity (class in spockbot.plugins.helpers.entities), 30

eye_pos (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 8, 28

E

edit_book() (spockbot.plugins.helpers.interact.InteractPlugin method), 9, 32

eid (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 27

eid (spockbot.plugins.helpers.entities.MCEntity attribute), 30

emit() (spockbot.plugins.core.event.EventPlugin method), 24

emit_open_window() (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34

emit_set_slot() (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34

enable_crypto() (spockbot.plugins.core.net.NetCore method), 3, 24

encode() (spockbot.mcp.mcpacket.Packet method), 20

encode() (spockbot.plugins.helpers.channels.ChannelsCore method), 5, 26

encrypt() (spockbot.plugins.core.net.AESCipher method), 24

EntitiesCore (class in spockbot.plugins.helpers.entities), 9, 29

F

fill() (spockbot.plugins.tools.smpmap.ChunkData method), 39

final_target (spockbot.plugins.helpers.movement.MovementCore attribute), 12, 34

find_by() (in module spockbot.mcpdata.utils), 17

find_slot() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33

find_slots() (spockbot.plugins.helpers.inventory.InventoryCore method), 11, 33

find_valid_nodes() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35

finish_digging() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

floor() (spockbot.vector.CartesianVector method), 42

FlowerPotData (class in spockbot.plugins.tools.smpmap), 40

flush() (spockbot.mcp.bbuff.BoundsBuffer method), 19

fmt (spockbot.mcp.nbt.TagByte attribute), 21

fmt (spockbot.mcp.nbt.TagDouble attribute), 21

fmt (spockbot.mcp.nbt.TagFloat attribute), 21

fmt (spockbot.mcp.nbt.TagInt attribute), [21](#)
 fmt (spockbot.mcp.nbt.TagLong attribute), [22](#)
 fmt (spockbot.mcp.nbt.TagShort attribute), [22](#)
 follow_path() (spockbot.plugins.helpers.movement.MovementPlugin
 method), [34](#)
 full_tasktrace (spockbot.plugins.tools.task.TaskFailed at-
 tribute), [41](#)

G

game_info (spockbot.plugins.helpers.clientinfo.ClientInfo
 attribute), [7](#), [28](#)
 GameInfo (class in spockbot.plugins.helpers.clientinfo),
[29](#)
 gen_block_set() (in module spock-
 bot.plugins.tools.collision), [37](#)
 get() (spockbot.plugins.tools.smpmap.BiomeData
 method), [38](#)
 get() (spockbot.plugins.tools.smpmap.ChunkData
 method), [39](#)
 get() (spockbot.plugins.tools.smpmap.ChunkDataNibble
 method), [39](#)
 get_any_recipe() (in module spockbot.mcdata.recipes),
[17](#)
 get_biome() (in module spockbot.mcdata.biomes), [15](#)
 get_biome() (spockbot.plugins.tools.smpmap.Dimension
 method), [39](#)
 get_block() (in module spockbot.mcdata.blocks), [16](#)
 get_block() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin
 method), [35](#)
 get_block() (spockbot.plugins.tools.smpmap.Dimension
 method), [39](#)
 get_block_entity_data() (spock-
 bot.plugins.tools.smpmap.Dimension method),
[39](#)
 get_block_slip() (spock-
 bot.plugins.helpers.physics.PhysicsPlugin
 method), [36](#)
 get_dict() (spockbot.mcdata.utils.Info method), [17](#)
 get_dict() (spockbot.mcdata.windows.Slot method), [18](#)
 get_dict() (spockbot.plugins.helpers.clientinfo.Position
 method), [29](#)
 get_dict() (spockbot.vector.Vector3 method), [43](#)
 get_item() (in module spockbot.mcdata.items), [16](#)
 get_item_or_block() (in module spockbot.mcdata), [19](#)
 get_light() (spockbot.plugins.tools.smpmap.Dimension
 method), [39](#)
 get_material() (in module spockbot.mcdata.materials), [16](#)
 get_mtv() (spockbot.plugins.helpers.physics.PhysicsPlugin
 method), [36](#)
 get_packet() (spockbot.mcdata.windows.BaseClick
 method), [17](#)
 get_packet() (spockbot.mcdata.windows.DropClick
 method), [18](#)

get_packet() (spockbot.mcdata.windows.SingleClick
 method), [18](#)
 get_plugin_settings() (spock-
 bot.plugins.core.settings.PloaderFetch
 method), [25](#)
 get_plugins() (spockbot.plugins.core.settings.PloaderFetch
 method), [25](#)
 get_settings() (in module spockbot.plugins.base), [42](#)
 get_shared_secret() (spock-
 bot.plugins.core.auth.AuthCore method),
[2](#), [23](#)
 get_username() (spockbot.plugins.core.auth.AuthCore
 method), [2](#), [23](#)
 global_type (spockbot.plugins.helpers.entities.GlobalEntity
 attribute), [30](#)
 GlobalEntity (class in spockbot.plugins.helpers.entities),
[30](#)

H

handle_attach_entity() (spock-
 bot.plugins.helpers.clientinfo.ClientInfoPlugin
 method), [28](#)
 handle_auth_error() (spock-
 bot.plugins.core.auth.AuthPlugin method),
[24](#)
 handle_block_change() (spock-
 bot.plugins.helpers.world.WorldPlugin
 method), [37](#)
 handle_chat() (spockbot.plugins.helpers.chat.ChatPlugin
 method), [27](#)
 handle_chunk_data() (spock-
 bot.plugins.helpers.world.WorldPlugin
 method), [37](#)
 handle_close_window() (spock-
 bot.plugins.helpers.inventory.InventoryPlugin
 method), [34](#)
 handle_comp() (spockbot.plugins.core.net.NetPlugin
 method), [25](#)
 handle_confirm_transaction() (spock-
 bot.plugins.helpers.inventory.InventoryPlugin
 method), [34](#)
 handle_destroy_entities() (spock-
 bot.plugins.helpers.entities.EntitiesPlugin
 method), [29](#)
 handle_disconnect() (spock-
 bot.plugins.core.net.NetPlugin method),
[25](#)
 handle_disconnect() (spock-
 bot.plugins.helpers.clientinfo.ClientInfoPlugin
 method), [28](#)
 handle_disconnect() (spock-
 bot.plugins.helpers.world.WorldPlugin
 method), [37](#)

handle_err()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_recv()	(spockbot.plugins.core.net.NetPlugin method), 25
handle_game_state()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_relative_move()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 29
handle_handshake()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_select_err()	(spockbot.plugins.core.net.NetPlugin method), 25
handle_held_item_change()	(spockbot.plugins.helpers.inventory.InventoryPlugin method), 34	handle_send()	(spockbot.plugins.core.net.NetPlugin method), 25
handle_hup()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_server_difficulty()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 29
handle_join_game()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_session_error()	(spockbot.plugins.core.auth.AuthPlugin method), 24
handle_join_game()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 29	handle_set_dict()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 29
handle_kill()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_set_slot()	(spockbot.plugins.helpers.inventory.InventoryPlugin method), 34
handle_login_disconnect()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_spawn_experience_orb()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_login_success()	(spockbot.plugins.core.net.NetPlugin method), 25	handle_spawn_global_entity()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_login_success()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_spawn_mob()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_map_chunk_bulk()	(spockbot.plugins.helpers.world.WorldPlugin method), 37	handle_spawn_object()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_multi_block_change()	(spockbot.plugins.helpers.world.WorldPlugin method), 37	handle_spawn_painting()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_new_dimension()	(spockbot.plugins.helpers.world.WorldPlugin method), 37	handle_spawn_player()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_open_window()	(spockbot.plugins.helpers.inventory.InventoryPlugin method), 34	handle_spawn_position()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 29
handle_player_abilities()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_time_update()	(spockbot.plugins.helpers.world.WorldPlugin method), 37
handle_player_list()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_unhandled()	(spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
handle_plugin_message()	(spockbot.plugins.helpers.channels.ChannelsPlugin method), 26	handle_update_block_entity()	(spockbot.plugins.helpers.world.WorldPlugin method), 37
handle_position_update()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 28	handle_update_health()	(spockbot.plugins.helpers.clientinfo.ClientInfoPlugin method), 29

[handle_update_sign\(\)](#) (spockbot.plugins.helpers.world.WorldPlugin method), 37
[handle_velocity\(\)](#) (spockbot.plugins.helpers.entities.EntitiesPlugin method), 30
[handle_window_items\(\)](#) (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34
[handle_window_prop\(\)](#) (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34
[hardness](#) (spockbot.mcdata.blocks.Block attribute), 15
[harvest_tools](#) (spockbot.mcdata.blocks.Block attribute), 15
[head_pitch](#) (spockbot.plugins.helpers.entities.MobEntity attribute), 30
[head_yaw](#) (spockbot.plugins.helpers.entities.MobEntity attribute), 30
[HeadData](#) (class in spockbot.plugins.tools.smpmap), 40
[health](#) (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 28
[hold_item\(\)](#) (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
[hotbar_slots](#) (spockbot.mcdata.windows.Window attribute), 18

I

[iadd\(\)](#) (spockbot.vector.CartesianVector method), 42
[iceil\(\)](#) (spockbot.vector.CartesianVector method), 42
[id](#) (spockbot.mcdata.blocks.Block attribute), 15
[id](#) (spockbot.mcdata.items.Item attribute), 16
[id](#) (spockbot.mcdata.recipes.RecipeItem attribute), 17
[id](#) (spockbot.mcp.nbt.Tag attribute), 20
[id](#) (spockbot.mcp.nbt.TagByte attribute), 21
[id](#) (spockbot.mcp.nbt.TagByteArray attribute), 21
[id](#) (spockbot.mcp.nbt.TagCompound attribute), 21
[id](#) (spockbot.mcp.nbt.TagDouble attribute), 21
[id](#) (spockbot.mcp.nbt.TagFloat attribute), 21
[id](#) (spockbot.mcp.nbt.TagInt attribute), 21
[id](#) (spockbot.mcp.nbt.TagIntArray attribute), 21
[id](#) (spockbot.mcp.nbt.TagList attribute), 22
[id](#) (spockbot.mcp.nbt.TagLong attribute), 22
[id](#) (spockbot.mcp.nbt.TagShort attribute), 22
[id](#) (spockbot.mcp.nbt.TagString attribute), 22
[idiv\(\)](#) (spockbot.vector.CartesianVector method), 42
[ifloor\(\)](#) (spockbot.vector.CartesianVector method), 42
[imul\(\)](#) (spockbot.vector.CartesianVector method), 43
[Info](#) (class in spockbot.mcdata.utils), 17
[ingredient_positions](#) (spockbot.mcdata.recipes.Recipe attribute), 16
[init\(\)](#) (spockbot.vector.BaseVector method), 42
[init\(\)](#) (spockbot.vector.Vector3 method), 43
[init\(\)](#) (spockbot.vector.YawPitch method), 43
[insert\(\)](#) (spockbot.mcp.nbt.TagByteArray method), 21
[insert\(\)](#) (spockbot.mcp.nbt.TagIntArray method), 21
[insert\(\)](#) (spockbot.mcp.nbt.TagList method), 22
[InteractPlugin](#) (class in spockbot.plugins.helpers.interact), 9, 31
[inv_data](#) (spockbot.mcdata.windows.Window attribute), 18
[inv_slots_preferred](#) (spockbot.plugins.helpers.inventory.InventoryCore attribute), 11, 33
[inv_type](#) (spockbot.mcdata.windows.Window attribute), 18
[invalidate\(\)](#) (spockbot.mcp.yggdrasil.YggdrasilCore method), 22
[inventory_slots](#) (spockbot.mcdata.windows.Window attribute), 18
[InventoryAsync](#) (class in spockbot.plugins.tools.inventory_async), 38
[InventoryCore](#) (class in spockbot.plugins.helpers.inventory), 11, 33
[InventoryPlugin](#) (class in spockbot.plugins.helpers.inventory), 34
[is_async](#) (spockbot.mcdata.windows.Slot attribute), 18
[is_moving](#) (spockbot.plugins.helpers.movement.MovementCore attribute), 12, 34
[isub\(\)](#) (spockbot.vector.CartesianVector method), 43
[Item](#) (class in spockbot.mcdata.items), 16
[iter_recipes\(\)](#) (in module spockbot.mcdata.recipes), 17
[iteritems\(\)](#) (spockbot.mcp.nbt.TagCompound method), 21
[itruediv\(\)](#) (spockbot.vector.CartesianVector method), 43

J

[java_hex_digest\(\)](#) (in module spockbot.plugins.core.auth), 24
[jump\(\)](#) (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 35
[jump_horse\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
[jump_vehicle\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

K

[keys\(\)](#) (spockbot.mcp.nbt.TagCompound method), 21
[kill\(\)](#) (spockbot.plugins.core.event.EventPlugin method), 24

L

[leave_bed\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
[length](#) (spockbot.plugins.tools.smpmap.BiomeData attribute), 38
[length](#) (spockbot.plugins.tools.smpmap.ChunkData attribute), 39

- ul style="list-style-type: none; padding-left: 0;">
- length (spockbot.plugins.tools.smpmap.ChunkDataNibble attribute), 39
- length (spockbot.plugins.tools.smpmap.ChunkDataShort attribute), 39
- load_translations() (spockbot.plugins.helpers.chat.ChatPlugin method), 27
- location (spockbot.plugins.helpers.entities.PaintingEntity attribute), 31
- login() (spockbot.mcp.yggdrasil.YggdrasilCore method), 22
- logout() (spockbot.mcp.yggdrasil.YggdrasilCore method), 22
- look() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
- look_at() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
- look_at_rel() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
- look_rel() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
- ## M
- make_slot_check() (in module spockbot.mcdata.windows), 18
 - MalformedFileError, 20
 - mapshort2id() (in module spockbot.plugins.tools.smpmap), 40
 - mark_dirty() (spockbot.mcdata.windows.BaseClick method), 17
 - matches() (spockbot.mcdata.windows.Slot method), 18
 - material (spockbot.mcdata.blocks.Block attribute), 16
 - MCEntity (class in spockbot.plugins.helpers.entities), 30
 - message (spockbot.plugins.tools.task.TaskFailed attribute), 40
 - meta (spockbot.mcdata.recipes.RecipeItem attribute), 17
 - metadata (spockbot.plugins.helpers.entities.MCEntity attribute), 30
 - metadata (spockbot.plugins.helpers.entities.MobEntity attribute), 30
 - metadata (spockbot.plugins.helpers.entities.PlayerEntity attribute), 31
 - mob_type (spockbot.plugins.helpers.entities.MobEntity attribute), 30
 - MobEntity (class in spockbot.plugins.helpers.entities), 30
 - mount_vehicle() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
 - move_angle() (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 35
 - move_target() (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 35
 - move_to_inventory() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
 - move_to_window() (spockbot.mcdata.windows.Slot method), 18
 - move_to_window() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38
 - move_vector() (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 36
 - MovementCore (class in spockbot.plugins.helpers.movement), 12, 34
 - MovementEntity (class in spockbot.plugins.helpers.entities), 31
 - MovementPlugin (class in spockbot.plugins.helpers.movement), 34
 - MTVTest (class in spockbot.plugins.tools.collision), 37
- ## N
- name (spockbot.mcdata.blocks.Block attribute), 16
 - name (spockbot.mcdata.items.Item attribute), 16
 - name (spockbot.mcdata.windows.Window attribute), 18
 - name (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 27
 - nbt (spockbot.plugins.helpers.entities.MCEntity attribute), 30
 - NetCore (class in spockbot.plugins.core.net), 3, 24
 - NetPlugin (class in spockbot.plugins.core.net), 25
 - new_dimension() (spockbot.plugins.helpers.world.WorldData method), 14, 36
 - new_ident() (spockbot.mcp.mcpacket.Packet method), 20
 - new_path() (spockbot.plugins.helpers.movement.MovementPlugin method), 34
 - norm() (spockbot.vector.CartesianVector method), 43
- ## O
- obj_data (spockbot.plugins.helpers.entities.ObjectEntity attribute), 31
 - obj_type (spockbot.plugins.helpers.entities.ObjectEntity attribute), 31
 - ObjectEntity (class in spockbot.plugins.helpers.entities), 31
 - on_error() (spockbot.plugins.tools.task.Task method), 40
 - on_error() (spockbot.plugins.tools.task.TaskCallback method), 40
 - on_event() (spockbot.plugins.tools.task.Task method), 40
 - on_ground (spockbot.plugins.helpers.entities.MovementEntity attribute), 31
 - on_success() (spockbot.mcdata.windows.BaseClick method), 17
 - on_success() (spockbot.plugins.tools.task.Task method), 40
 - on_success() (spockbot.plugins.tools.task.TaskCallback method), 40

open_inventory() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

P

pack() (in module spockbot.mcp.datautils), 19
 pack() (spockbot.plugins.tools.smpmap.ChunkData method), 39
 pack_fixed_point() (in module spockbot.mcp.datautils), 19
 pack_metadata() (in module spockbot.mcp.datautils), 19
 pack_position() (in module spockbot.mcp.datautils), 19
 pack_slot() (in module spockbot.mcp.datautils), 19
 pack_varint() (in module spockbot.mcp.datautils), 19
 pack_varlong() (in module spockbot.mcp.datautils), 19
 Packet (class in spockbot.mcp.mcpacket), 20
 PacketDecodeFailure, 20
 PaintingEntity (class in spockbot.plugins.helpers.entities), 31
 parse_response() (spockbot.plugins.tools.task.Task method), 40
 parse_with_1_extra() (in module spockbot.plugins.helpers.chat), 27
 password (spockbot.plugins.core.auth.AuthCore attribute), 2, 23
 Path (class in spockbot.plugins.helpers.pathfinding), 35
 path_cb() (spockbot.plugins.helpers.movement.MovementPlugin method), 35
 pathfind() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
 PathfindingCore (class in spockbot.plugins.helpers.pathfinding), 13, 35
 PathfindingPlugin (class in spockbot.plugins.helpers.pathfinding), 35
 PathNode (class in spockbot.plugins.helpers.pathfinding), 35
 persistent_slots (spockbot.mcdata.windows.Window attribute), 18
 physics_tick() (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
 PhysicsCore (class in spockbot.plugins.helpers.physics), 13, 35
 PhysicsPlugin (class in spockbot.plugins.helpers.physics), 36
 pitch (spockbot.plugins.helpers.entities.MovementEntity attribute), 31
 pitch (spockbot.vector.YawPitch attribute), 43
 pl_announce (spockbot.plugins.core.auth.AuthPlugin attribute), 24
 pl_announce (spockbot.plugins.core.event.EventPlugin attribute), 24
 pl_announce (spockbot.plugins.core.net.NetPlugin attribute), 25

pl_announce (spockbot.plugins.core.settings.SettingsPlugin attribute), 25
 pl_announce (spockbot.plugins.core.taskmanager.TaskManager attribute), 26
 pl_announce (spockbot.plugins.helpers.channels.ChannelsPlugin attribute), 26
 pl_announce (spockbot.plugins.helpers.chat.ChatPlugin attribute), 27
 pl_announce (spockbot.plugins.helpers.clientinfo.ClientInfoPlugin attribute), 29
 pl_announce (spockbot.plugins.helpers.craft.CraftPlugin attribute), 8, 29
 pl_announce (spockbot.plugins.helpers.entities.EntitiesPlugin attribute), 30
 pl_announce (spockbot.plugins.helpers.interact.InteractPlugin attribute), 10, 32
 pl_announce (spockbot.plugins.helpers.inventory.InventoryPlugin attribute), 34
 pl_announce (spockbot.plugins.helpers.movement.MovementPlugin attribute), 35
 pl_announce (spockbot.plugins.helpers.pathfinding.PathfindingPlugin attribute), 35
 pl_announce (spockbot.plugins.helpers.physics.PhysicsPlugin attribute), 36
 pl_announce (spockbot.plugins.helpers.world.WorldPlugin attribute), 37
 pl_announce() (in module spockbot.plugins.base), 42
 pl_event() (in module spockbot.plugins.base), 42
 place_block() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
 place_sign() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
 player_list (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 8, 28
 PlayerEntity (class in spockbot.plugins.helpers.entities), 31
 PlayerHealth (class in spockbot.plugins.helpers.clientinfo), 29
 PlayerListItem (class in spockbot.plugins.helpers.clientinfo), 29
 PlayerPosition (class in spockbot.plugins.helpers.clientinfo), 29
 PloaderFetch (class in spockbot.plugins.core.settings), 25
 PluginBase (class in spockbot.plugins.base), 42
 PluginLoader (class in spockbot.plugins.loader), 42
 Position (class in spockbot.plugins.helpers.clientinfo), 29
 position (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 8, 28
 pretty_tree() (spockbot.mcp.nbt.Tag method), 20
 pretty_tree() (spockbot.mcp.nbt.TagCompound method), 21
 pretty_tree() (spockbot.mcp.nbt.TagList method), 22
 prev_error (spockbot.plugins.tools.task.TaskFailed attribute), 41

provides() (spockbot.plugins.loader.PluginLoader method), 42
 push() (spockbot.plugins.core.net.NetCore method), 3, 24
 push_packet() (spockbot.plugins.core.net.NetCore method), 3, 24

R

raycast_bbox() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35
 read() (spockbot.mcp.bbuff.BoundBuffer method), 19
 read_packet() (spockbot.plugins.core.net.NetCore method), 3, 24
 Recipe (class in spockbot.mcddata.recipes), 16
 RecipeItem (class in spockbot.mcddata.recipes), 16
 recv() (spockbot.mcp.bbuff.BoundBuffer method), 19
 reformat_item() (in module spockbot.mcddata.recipes), 17
 reformat_shape() (in module spockbot.mcddata.recipes), 17
 refresh() (spockbot.mcp.yggdrasil.YggdrasilCore method), 23
 reg_event_handler() (spockbot.plugins.core.event.EventPlugin method), 24
 register() (spockbot.plugins.tools.task.Task method), 40
 render_chat() (spockbot.plugins.helpers.chat.ChatPlugin method), 27
 requires (spockbot.plugins.base.PluginBase attribute), 42
 requires (spockbot.plugins.core.auth.AuthPlugin attribute), 24
 requires (spockbot.plugins.core.net.NetPlugin attribute), 25
 requires (spockbot.plugins.core.taskmanager.TaskManager attribute), 26
 requires (spockbot.plugins.core.ticker.TickerPlugin attribute), 26
 requires (spockbot.plugins.helpers.channels.ChannelsPlugin attribute), 26
 requires (spockbot.plugins.helpers.chat.ChatPlugin attribute), 27
 requires (spockbot.plugins.helpers.clientinfo.ClientInfoPlugin attribute), 29
 requires (spockbot.plugins.helpers.craft.CraftPlugin attribute), 8, 29
 requires (spockbot.plugins.helpers.entities.EntitiesPlugin attribute), 30
 requires (spockbot.plugins.helpers.interact.InteractPlugin attribute), 10, 32
 requires (spockbot.plugins.helpers.inventory.InventoryPlugin attribute), 34
 requires (spockbot.plugins.helpers.movement.MovementPlugin attribute), 35
 requires (spockbot.plugins.helpers.pathfinding.PathfindingPlugin attribute), 35
 requires (spockbot.plugins.helpers.physics.PhysicsPlugin attribute), 36
 requires (spockbot.plugins.helpers.start.StartPlugin attribute), 36
 requires (spockbot.plugins.helpers.world.WorldPlugin attribute), 37
 requires() (spockbot.plugins.loader.PluginLoader method), 42
 reset() (spockbot.plugins.core.net.NetCore method), 3, 25
 reset() (spockbot.plugins.helpers.clientinfo.ClientInfo method), 8, 28
 reset() (spockbot.plugins.helpers.world.WorldData method), 14, 36
 reset_sock() (spockbot.plugins.core.net.NetPlugin method), 25
 resume_physics() (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
 revert() (spockbot.mcp.bbuff.BoundBuffer method), 19
 rpitch (spockbot.vector.YawPitch attribute), 43
 run() (spockbot.plugins.tools.task.Task method), 40
 run_continuous() (spockbot.plugins.core.event.EventPlugin method), 24
 run_once() (spockbot.plugins.core.event.EventPlugin method), 24
 run_task() (spockbot.plugins.core.taskmanager.TaskManager method), 26
 ryaw (spockbot.vector.YawPitch attribute), 43

S

save() (spockbot.mcp.bbuff.BoundBuffer method), 19
 select_active_slot() (spockbot.plugins.helpers.inventory.InventoryCore method), 12, 33
 send() (spockbot.plugins.helpers.channels.ChannelsCore method), 5, 26
 send_click() (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34
 send_session_auth() (spockbot.plugins.core.auth.AuthCore method), 2, 23
 set() (spockbot.plugins.helpers.pathfinding.PathNode method), 35
 set() (spockbot.plugins.tools.smpmap.BiomeData method), 38
 set() (spockbot.plugins.tools.smpmap.ChunkData method), 39
 set() (spockbot.plugins.tools.smpmap.ChunkDataNibble method), 39
 set_auth_token() (spockbot.plugins.core.auth.AuthCore method), 2, 23
 set_biome() (spockbot.plugins.tools.smpmap.Dimension method), 39

[set_block\(\)](#) (spockbot.plugins.tools.smpmap.Dimension method), 39
[set_block_entity_data\(\)](#) (spockbot.plugins.tools.smpmap.Dimension method), 39
[set_client_token\(\)](#) (spockbot.plugins.core.auth.AuthCore method), 2, 23
[set_comp_state\(\)](#) (spockbot.plugins.core.net.NetCore method), 3, 25
[set_dict\(\)](#) (spockbot.mcdata.utils.Info method), 17
[set_dict\(\)](#) (spockbot.vector.Vector3 method), 43
[set_light\(\)](#) (spockbot.plugins.tools.smpmap.Dimension method), 40
[set_password\(\)](#) (spockbot.plugins.core.auth.AuthCore method), 2, 23
[set_proto_state\(\)](#) (spockbot.plugins.core.net.NetCore method), 3, 25
[set_slot\(\)](#) (spockbot.plugins.helpers.inventory.InventoryPlugin method), 34
[set_username\(\)](#) (spockbot.plugins.core.auth.AuthCore method), 2, 23
[SettingsPlugin](#) (class in spockbot.plugins.core.settings), 25
[shared_secret](#) (spockbot.plugins.core.auth.AuthCore attribute), 2, 23
[sign_book\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
[SignData](#) (class in spockbot.plugins.tools.smpmap), 40
[signout\(\)](#) (spockbot.mcp.yggdrasil.YggdrasilCore method), 23
[SingleClick](#) (class in spockbot.mcdata.windows), 18
[skip_physics\(\)](#) (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36
[slipperiness](#) (spockbot.mcdata.blocks.Block attribute), 16
[Slot](#) (class in spockbot.mcdata.windows), 18
[SlotCursor](#) (class in spockbot.mcdata.windows), 18
[snake_case\(\)](#) (in module spockbot.mcdata.utils), 17
[sneak\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
[spawn_position](#) (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 28
[SpawnerData](#) (class in spockbot.plugins.tools.smpmap), 40
[speed_x](#) (spockbot.plugins.helpers.entities.ObjectEntity attribute), 31
[speed_y](#) (spockbot.plugins.helpers.entities.ObjectEntity attribute), 31
[speed_z](#) (spockbot.plugins.helpers.entities.ObjectEntity attribute), 31
[split_words\(\)](#) (in module spockbot.mcdata.utils), 17
[spockbot](#) (module), 43
[spockbot.mcdata](#) (module), 19
[spockbot.mcdata.biomes](#) (module), 15
[spockbot.mcdata.blocks](#) (module), 15
[spockbot.mcdata.constants](#) (module), 16
[spockbot.mcdata.items](#) (module), 16
[spockbot.mcdata.materials](#) (module), 16
[spockbot.mcdata.recipes](#) (module), 16
[spockbot.mcdata.utils](#) (module), 17
[spockbot.mcdata.windows](#) (module), 17
[spockbot.mcp](#) (module), 23
[spockbot.mcp.bbuff](#) (module), 19
[spockbot.mcp.datautils](#) (module), 19
[spockbot.mcp.mcpacket](#) (module), 20
[spockbot.mcp.nbt](#) (module), 20
[spockbot.mcp.yggdrasil](#) (module), 22
[spockbot.plugins](#) (module), 42
[spockbot.plugins.base](#) (module), 42
[spockbot.plugins.core](#) (module), 26
[spockbot.plugins.core.auth](#) (module), 1, 23
[spockbot.plugins.core.event](#) (module), 2, 24
[spockbot.plugins.core.net](#) (module), 2, 24
[spockbot.plugins.core.settings](#) (module), 3, 25
[spockbot.plugins.core.taskmanager](#) (module), 3, 26
[spockbot.plugins.core.ticker](#) (module), 4, 26
[spockbot.plugins.helpers](#) (module), 37
[spockbot.plugins.helpers.channels](#) (module), 4, 26
[spockbot.plugins.helpers.chat](#) (module), 5, 27
[spockbot.plugins.helpers.clientinfo](#) (module), 6, 27
[spockbot.plugins.helpers.craft](#) (module), 8, 29
[spockbot.plugins.helpers.entities](#) (module), 9, 29
[spockbot.plugins.helpers.interact](#) (module), 9, 31
[spockbot.plugins.helpers.inventory](#) (module), 11, 33
[spockbot.plugins.helpers.movement](#) (module), 12, 34
[spockbot.plugins.helpers.pathfinding](#) (module), 13, 35
[spockbot.plugins.helpers.physics](#) (module), 13, 35
[spockbot.plugins.helpers.start](#) (module), 14, 36
[spockbot.plugins.helpers.world](#) (module), 14, 36
[spockbot.plugins.loader](#) (module), 42
[spockbot.plugins.tools](#) (module), 41
[spockbot.plugins.tools.collision](#) (module), 37
[spockbot.plugins.tools.event](#) (module), 37
[spockbot.plugins.tools.inventory_async](#) (module), 38
[spockbot.plugins.tools.smpmap](#) (module), 38
[spockbot.plugins.tools.task](#) (module), 40
[spockbot.vector](#) (module), 42
[sprintf\(\)](#) (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
[sprintf\(\)](#) (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 36
[stack_size](#) (spockbot.mcdata.blocks.Block attribute), 16
[stack_size](#) (spockbot.mcdata.items.Item attribute), 16
[stacks_with\(\)](#) (spockbot.mcdata.windows.Slot method), 18
[start\(\)](#) (spockbot.plugins.helpers.start.StartPlugin method), 36

start_digging() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

start_path() (spockbot.plugins.helpers.pathfinding.PathfindingPlugin method), 35

start_session() (spockbot.plugins.core.auth.AuthCore method), 2, 24

start_session_and_connect() (spockbot.plugins.helpers.start.StartPlugin method), 36

start_tickers() (spockbot.plugins.core.ticker.TickerPlugin method), 26

StartPlugin (class in spockbot.plugins.helpers.start), 36

status (spockbot.plugins.helpers.entities.MCEntity attribute), 30

steer_vehicle() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

stop() (spockbot.plugins.helpers.movement.MovementCore method), 12, 34

store_or_drop() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38

suspend_physics() (spockbot.plugins.helpers.physics.PhysicsPlugin method), 36

swap_slots() (spockbot.mcdata.windows.BaseClick method), 18

swap_slots() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38

swing_arm() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

T

Tag (class in spockbot.mcp.nbt), 20

tag_info() (spockbot.mcp.nbt.Tag method), 20

TagByte (class in spockbot.mcp.nbt), 20

TagByteArray (class in spockbot.mcp.nbt), 21

TagCompound (class in spockbot.mcp.nbt), 21

TagDouble (class in spockbot.mcp.nbt), 21

TagFloat (class in spockbot.mcp.nbt), 21

TagInt (class in spockbot.mcp.nbt), 21

TagIntArray (class in spockbot.mcp.nbt), 21

TagList (class in spockbot.mcp.nbt), 22

TagLong (class in spockbot.mcp.nbt), 22

TagShort (class in spockbot.mcp.nbt), 22

TagString (class in spockbot.mcp.nbt), 22

Task (class in spockbot.plugins.tools.task), 40

TaskCallback (class in spockbot.plugins.tools.task), 40

TaskFailed, 40

TaskManager (class in spockbot.plugins.core.taskmanager), 26

tasktrace (spockbot.plugins.tools.task.Task attribute), 40

tasktrace (spockbot.plugins.tools.task.TaskFailed attribute), 40

tell() (spockbot.mcp.bbuff.BoundsBuffer method), 19

tick() (spockbot.plugins.core.net.NetPlugin method), 25

TickerPlugin (class in spockbot.plugins.core.ticker), 26

tingPlugin (spockbot.plugins.helpers.entities.PaintingEntity attribute), 31

total_ingredient_amounts (spockbot.mcdata.recipes.Recipe attribute), 16

total_stored() (spockbot.plugins.helpers.inventory.InventoryCore method), 12, 34

transfer() (spockbot.mcdata.windows.BaseClick method), 18

transfer_slots() (spockbot.plugins.tools.inventory_async.InventoryAsync method), 38

trunc() (spockbot.vector.CartesianVector method), 43

trunc() (spockbot.plugins.tools.smpmap.ChunkData attribute), 39

trunc() (spockbot.plugins.tools.smpmap.ChunkDataShort attribute), 39

U

uncenter_position() (in module spockbot.plugins.tools.collision), 37

unit_vector() (spockbot.vector.YawPitch method), 43

unmount_vehicle() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

unpack() (in module spockbot.mcp.datautils), 19

unpack() (spockbot.plugins.tools.smpmap.ChunkColumn method), 39

unpack() (spockbot.plugins.tools.smpmap.ChunkData method), 39

unpack_bulk() (spockbot.plugins.tools.smpmap.Dimension method), 40

unpack_column() (spockbot.plugins.tools.smpmap.Dimension method), 40

unpack_fixed_point() (in module spockbot.mcp.datautils), 19

unpack_metadata() (in module spockbot.mcp.datautils), 19

unpack_position() (in module spockbot.mcp.datautils), 19

unpack_slot() (in module spockbot.mcp.datautils), 19

unpack_slots_list() (in module spockbot.plugins.tools.inventory_async), 38

unpack_varint() (in module spockbot.mcp.datautils), 20

unpack_varlong() (in module spockbot.mcp.datautils), 20

unreg_event_handler() (spockbot.plugins.core.event.EventPlugin method), 24

unsneak() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

unsprint() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32

update_fmt() (spockbot.mcp.nbt.TagIntArray method), 22
update_time() (spockbot.plugins.helpers.world.WorldData method), 15, 37
use_bucket() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
use_entity() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 32
username (spockbot.plugins.core.auth.AuthCore attribute), 2, 24
uuid (spockbot.plugins.helpers.clientinfo.ClientInfo attribute), 7, 28
uuid (spockbot.plugins.helpers.entities.PlayerEntity attribute), 31

V

validate() (spockbot.mcp.yggdrasil.YggdrasilCore method), 23
valuestr() (spockbot.mcp.nbt.Tag method), 20
valuestr() (spockbot.mcp.nbt.TagByteArray method), 21
valuestr() (spockbot.mcp.nbt.TagCompound method), 21
valuestr() (spockbot.mcp.nbt.TagIntArray method), 22
valuestr() (spockbot.mcp.nbt.TagList method), 22
variations (spockbot.mcdata.blocks.Block attribute), 16
variations (spockbot.mcdata.items.Item attribute), 16
Vector3 (class in spockbot.vector), 43
velocity_x (spockbot.plugins.helpers.entities.MobEntity attribute), 30
velocity_y (spockbot.plugins.helpers.entities.MobEntity attribute), 30
velocity_z (spockbot.plugins.helpers.entities.MobEntity attribute), 30
yaw (spockbot.plugins.helpers.entities.MovementEntity attribute), 31
yaw (spockbot.vector.Vector3 attribute), 43
yaw_pitch (spockbot.vector.Vector3 attribute), 43
YawPitch (class in spockbot.vector), 43
ygg_url (spockbot.mcp.yggdrasil.YggdrasilCore attribute), 23
ygg_version (spockbot.mcp.yggdrasil.YggdrasilCore attribute), 23
YggdrasilCore (class in spockbot.mcp.yggdrasil), 22

Z

z (spockbot.plugins.helpers.entities.ExpEntity attribute), 30
z (spockbot.plugins.helpers.entities.GlobalEntity attribute), 30
z (spockbot.plugins.helpers.entities.MovementEntity attribute), 31
z (spockbot.vector.Vector3 attribute), 43
zero() (spockbot.vector.CartesianVector method), 43

W

walk() (spockbot.plugins.helpers.physics.PhysicsCore method), 13, 36
whisper() (spockbot.plugins.helpers.chat.ChatCore method), 6, 27
Window (class in spockbot.mcdata.windows), 18
window_slots (spockbot.mcdata.windows.Window attribute), 18
with_error() (spockbot.plugins.tools.task.TaskFailed method), 41
WorldData (class in spockbot.plugins.helpers.world), 14, 36
WorldPlugin (class in spockbot.plugins.helpers.world), 37
write() (spockbot.mcp.bbuff.BoundBuffer method), 19
write_book() (spockbot.plugins.helpers.interact.InteractPlugin method), 10, 33

X

x (spockbot.plugins.helpers.entities.ExpEntity attribute), 30