# ps-client

Sep 21, 2021

# Contents:

# CHAPTER 1

## Connection class

**class** psclient.**Connection**(*username*, *password*, *websocket*, *chatlogger*, *loglevel*)

Bases: object

Represents a connection to Pokemon Showdown

**You should NOT be constructing this class yourself.** Instead, use *psclient.connect()*.

**Parameters**

- **username** (*string*) – the username to log in to PS! with
- **password** (*string*) – the password for the PS! username provided
- **websocket** (*object*) – the websocket of the server to connect to.
- **chatlogger** (*object*) – a chatlogger, whose *handleMessage()* method will be called on each message.
- **loglevel** (*int*) – the level of logging (to stdout / stderr). Higher means more verbose.

**Variables**

- **roomList** (*set*) – a set of all the known *Room* objects
- **userList** (*dictionary*) – a dictionary mapping all known *User* objects to lists of room IDs
- **password** (*string*) – the password to use to log into PS
- **loglevel** (*int*) – the level of logging
- **lastSentTime** (*int*) – the timestamp at which the most recent message was sent
- **this** (User) – an *User* object referring to the user who's logged in
- **isLoggedIn** (*bool*) – True if the connection represents a logged-in user and False otherwise

**getMessage**() → psclient.Message

Gets a message from the websocket and parses it as needed

This is a blocking method if awaited, but you can use asyncio.wait_for() safely.

**Returns** the message, or None if no message was received

**Return type** *Message* or None

**getRoom**(*name*)

Gets the `Room` object corresponding to an ID

**Parameters** **id** (*string in ID format*) – the room ID (in ID format from `toID()`)

**Returns** a `Room` object with the given ID

**Return type** *Room*

**getUser**(*userid*)

Gets the `User` object for a given ID

**Parameters** **userid** (*string that is an ID*) – the ID of the user to search for

**Returns** the user with the given ID

**Return type** *User* or None

**getUserRooms**(*user*)

Gets a set of the IDs (not objects) of the rooms that the user is in.

**Parameters** **user** (`User`) – the user

**Returns**

**Return type** set or None

**handleMessage**(*rawMessage: Union[str, bytes]*) → psclient.Message

Handles incoming raw messages

**Parameters** **rawMessage** (*Union[str, bytes]*) – the raw message from the websocket

**Returns** the parsed message

**Return type** *Message*

**log**(*message*)

Logs a message to the console according to *LOGLEVEL*

**Parameters** **message** (*string*) – the message to be logged, beginning with E:, W:, I:, or DEBUG:

**login**(*challstr*)

Logs into Pokemon Showdown

**Parameters** **challstr** (*string*) – the challstr to use to log in

**messages**() → AsyncGenerator[psclient.Message, None]

An async generator yielding messages from the websocket

**Yields** *Message* – a processed messages

**sayIn**(*room*, *message*)

Sends a message to a room.

**Parameters**

- **room** (`Room`) – the room to send the message to
- **message** (*string*) – the message to send

**send** (*message*)
Sends a message

**Parameters** **message** (*string*) – the message to send

**userJoinedRoom** (*user*, *room*)
Handles a user joining a room

**Parameters**

- **user** (User) – the user who joined
- **room** (Room) – the room they joined

**userLeftRoom** (*user*, *room*)
Handles a user leaving a room

**Parameters**

- **user** (User) – the user who joined
- **room** (Room) – the room they joined

**waitForLogin** ()
Waits for a challstr and then logs in.

May block infinitely if the server never sends a **|challstr|**, so using asyncio.wait_for() is advised.

**whisper** (*userid*, *message*)
PMs a message to a user

**Parameters**

- **userid** (*string in ID format*) – the user to PM
- **message** (*string*) – the message to PM

psclient.**connect** (*username: str*, *password: str*, *url='wss://sim3.psim.us/showdown/websocket'*, *chatlogger=None*, *loglevel=2*) → psclient.Connection
Creates a new connection to a PS server, and logs in

**Parameters**

- **username** (*str*) – the username to use for the connection
- **password** (*str*) – the password for the username
- **url** (*str, optional*) – the URL of the server to connect to. Defaults to "wss://sim3.psim.us/showdown/websocket".
- **chatlogger** (Chatlogger, *optional*) – a ps-client compatible chatlogger to use. Defaults to None.
- **loglevel** (*int, optional*) – the level of logging; higher is more verbose. Defaults to 2.

**Returns** an object representing the connection

**Return type** *Connection*

# CHAPTER 2

## Room class

**class** psclient.**Room**(*name*, *connection*)

Bases: object

Represents a room on Pokemon Showdown

**Parameters**

- **name** (*string*) – the name of the room that the *Room* object represents (can include spaces/caps)
- **connection** (Connection) – the *Connection* object to use to connect to the room

**Variables**

- **connection** (Connection) – the *Connection* object to use to connect to the room
- **id** (*string that is an ID*) – the room's ID
- **auth** (*dictionary*) – a dictionary containing the room's auth

**join**()

Joins the room

**leave**()

Leaves the room

**say**(*message*)

Sends a message to the room

**Parameters** **message** (*string*) – the message to send

**updateAuth**(*authDict*)

Updates the auth list for the room based on the given auth dictionary

**Parameters** **authDict** (*dictionary*) – dictionary of the changes to the auth list

**usersWithRankGEQ**(*rank*)

Gets a set of userids of the roomauth whose room rank is greater than or equal to a certain rank

**Parameters** **rank** (*string*) – the minimum rank

**Returns** a set of userids for the roomauth whose room rank is greater than or equal to the given rank

**Return type** set

# CHAPTER 3

# User class

**class** psclient.**User**(*name*, *connection*)

Bases: object

Represents a user on Pokemon Showdown

**Parameters**

- **name** (*string*) – the username
- **connection** (Connection) – the connection to access PS with

**Variables**

- **name** (*string*) – the username
- **connection** (Connection) – the connection to access PS with
- **id** (*string that is an ID*) – the user's ID

**PM**(*message*)

PMs the user the given message

**Parameters** **message** (*string*) – the message to PM the user

**canUseHTML**(*room*)

Checks if the user can use HTML

**Parameters** **room** (Room) – the room where the action is taking place

**Returns** True if the user can use HTML and False otherwise

**Return type** bool

# CHAPTER 4

# Message class

**class** psclient.**Message**(*raw*, *connection*)

Bases: object

Represents a message sent on Pokemon Showdown

**Parameters**

- **raw** (*string*) – the raw data of the message
- **connection** (Connection) – the connection the message was recieved on

**Variables**

- **sender** (User *or* None) – the user who sent the message
- **room** (Room *or* None) – the room the message was sent in
- **body** (*string or None*) – the body of the message
- **time** (*string or None*) – the UNIX timestamp of the message
- **type** (*string or None*) – the type of the message (chat, pm, etc)
- **challstr** (*string or None*) – the challstr, if the message contains one
- **senderName** (*string or None*) – the username of the user who sent the message
- **raw** (*string*) – the raw message
- **connection** (Connection) – the connection the message was recieved on

**respondHTML**(*html*)

Responds to the message with a HTML box, in a room or in PMs

If the user cannot broadcast and the command wasn't in PMs or it's not a message that can be responded to, does nothing

**Parameters** **html** (*string*) – the html to be sent

# CHAPTER 5

## chatlog module

a sample chatlogger included with ps-client

**class** chatlog.**Chatlogger**(*path*)

Bases: object

Class for logging chat

**Parameters** **path** (*string*) – the path to the logging directory

**formatData**(*data*, *isHTML=False*)

Formats data to text

**Parameters**

- **data** (*string of form userid|time|type|senderName|body*) – the data
- **isHTML** (*bool, optional*) – Whether to format as HTML. Defaults to False.

**Returns** a human-readable version of the message

**Return type** string

**formatMessage**(*message*)

Formats a message for logging in the data format userid|time|type|senderName|body

**Parameters** **message** (Message) – the message to format

**Returns** the formatted message

**Return type** (string)

**getFile**(*roomID*, *perms*)

Returns a file object corresponding to the room's chatlog file.

**Parameters**

- **roomID** (*string that is an ID*) – the room
- **perms** (*string*) – the file perms (for example, 'r' or 'w')

**Returns** a file for the log file for that room and day

**Return type** File

**handleMessage**(*message*)

Handles logging a message to chatlogs

**Parameters** **message** (Message) – the Message

**path = None**

the path to log chat to

**search**(*roomID=”*, *userID=”*, *keyword=”*, *includeJoins=False*)

Searches chatlogs

**Parameters**

- **roomID** (*str, optional*) – The ID of the room to search in. Defaults to “”.
- **userID** (*str, optional*) – The ID of the user whose messages are being searched for. Defaults to “”.
- **keyword** (*str, optional*) – [description]. Defaults to “”.

**Returns** a dictionary of matched messages (formatted as `{date (string): [userid|time|type|senderName|body] (list of day's results)}`)

**Return type** dictionary

CHAPTER 6

# tests module

Help for tests

CHAPTER 7

# Indices and tables

- genindex
- search

# Python Module Index

## c



## t

# Index