

---

**ps-client**

**Sep 18, 2021**



---

## Contents:

---

<b>1</b>	<b>PSClient class</b>	<b>1</b>
<b>2</b>	<b>PSConnection class</b>	<b>3</b>
<b>3</b>	<b>Room class</b>	<b>7</b>
<b>4</b>	<b>User class</b>	<b>9</b>
<b>5</b>	<b>Message class</b>	<b>11</b>
<b>6</b>	<b>chatlog module</b>	<b>13</b>
<b>7</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



# CHAPTER 1

---

## PSClient class

---

```
class psclient.PSClient (connection)
```

Bases: object

A Pokemon Showdown client

**Parameters** **connection** (*PSConnection*) – the connection to use to connect the client to PS!

**Variables** **connection** (*PSConnection*) – the connection to PS!

```
connect ()
```

Runs the client, logging in and connecting.

Note that this function is blocking: statements after this function is called will not be executed.

However, your program will keep running with the `PSConnection.onParsedMessage` and `onOpenThread` attributes.



## CHAPTER 2

---

### PSConnection class

---

```
class psclient.PSConnection(username, password, onParsedMes-
                             sage=None, onOpenThread=None,
                             url='ws://sim.smogon.com:8000/showdown/websocket', chat-
                             logger=None, loglevel=1)
```

Bases: object

Represents a connection to Pokemon Showdown

#### Parameters

- **username** (*string*) – the username to log in to PS! with
- **password** (*string*) – the password for the PS! username provided
- **onParsedMessage** (*function, optional*) – a function that will be called each time a message is recieved the only argument passed in is the parsed *Message* object
- **onOpenThread** (*function, optional*) – a function that will run in its own thread once the socket is open, with the *PSConnection* as an argument. Defaults to None.
- **url** (*str, optional*) – the URL of the websocket of the server to connect to. Defaults to `ws://sim.smogon.com:8000/showdown/websocket`.
- **chatlogger** (*object, optional*) – a chatlogger, whose *handleMessage()* method will be called on each message. Defaults to None.
- **loglevel** (*int, optional*) – the level of logging (to stdout / stderr). Defaults to 1. 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
- **onParsedMessage** (*function*) – a function that will be called each time a message is recieved the only argument passed in is the parsed *Message* object
- **onOpenThread** (*function*) – a function that will run in its own thread once the socket is open, with the *PSConnection* as an *\*args* argument
- **isLoggedInIn** (*bool*) – True if the connection represents a logged-in user and False otherwise

**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

**login** (*challstr*)

Logs into Pokemon Showdown

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

**onClose** ()

Logs when the connection closes

**onError** (*error*)

Handles errors on the websocket

**Parameters** **{string? probably}** (*error*) – the error

**onMessage** (*rawMessage*)

Handles new messages from the websocket, creating a *Message* object

**Parameters** **rawMessage** (*string*) – the raw message data

**onOpen** ()

Logs when the websocket is opened

**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

**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



## CHAPTER 3

---

### 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** (*PSConnection*) – the *PSConnection* object to use to connect to the room

#### Variables

- **connection** (*PSConnection*) – the *PSConnection* 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

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

Bases: `object`

Represents a user on Pokemon Showdown

### Parameters

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

### Variables

- **name** (*string*) – the username
- **connection** (`PSConnection`) – 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

**can** (*action*, *room*)

Checks if the user may perform an action

### Parameters

- **action** (*string*) – the action (one of *wall* or *html*)
- **room** (`Room`) – the room where the action is taking place

**Returns** `True` if the user can do the action and `False` otherwise

**Return type** `bool`



---

## 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** (`PSConnection`) – 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** (`PSConnection`) – 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 6

---

### 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 7

---

### Indices and tables

---

- `genindex`
- `search`



### C

chatlog, [13](#)



## C

`can()` (*psclient.User method*), 9  
`chatlog` (*module*), 13  
`Chatlogger` (*class in chatlog*), 13  
`connect()` (*psclient.PSClient method*), 1

## F

`formatData()` (*chatlog.Chatlogger method*), 13  
`formatMessage()` (*chatlog.Chatlogger method*), 13

## G

`getFile()` (*chatlog.Chatlogger method*), 13  
`getRoom()` (*psclient.PSConnection method*), 4  
`getUser()` (*psclient.PSConnection method*), 4  
`getUserRooms()` (*psclient.PSConnection method*), 4

## H

`handleMessage()` (*chatlog.Chatlogger method*), 14

## J

`join()` (*psclient.Room method*), 7

## L

`leave()` (*psclient.Room method*), 7  
`login()` (*psclient.PSConnection method*), 4

## M

`Message` (*class in psclient*), 11

## O

`onClose()` (*psclient.PSConnection method*), 4  
`onError()` (*psclient.PSConnection method*), 4  
`onMessage()` (*psclient.PSConnection method*), 4  
`onOpen()` (*psclient.PSConnection method*), 4

## P

`path` (*chatlog.Chatlogger attribute*), 14  
`PM()` (*psclient.User method*), 9

`PSClient` (*class in psclient*), 1  
`PSConnection` (*class in psclient*), 3

## R

`respondHTML()` (*psclient.Message method*), 11  
`Room` (*class in psclient*), 7

## S

`say()` (*psclient.Room method*), 7  
`sayIn()` (*psclient.PSConnection method*), 4  
`search()` (*chatlog.Chatlogger method*), 14  
`send()` (*psclient.PSConnection method*), 5

## U

`updateAuth()` (*psclient.Room method*), 7  
`User` (*class in psclient*), 9  
`userJoinedRoom()` (*psclient.PSConnection method*), 5  
`userLeftRoom()` (*psclient.PSConnection method*), 5  
`usersWithRankGEQ()` (*psclient.Room method*), 7

## W

`whisper()` (*psclient.PSConnection method*), 5