
easy_{*tcp*}*Documentation*

Release unknown

nimpsch

Mar 08, 2020

Contents

1	Contents	3
1.1	License	3
1.2	Contributors	3
1.3	Changelog	3
1.4	easy_tcp	4
2	Indices and tables	9
	Python Module Index	11
	Index	13

This is the documentation of **easy_tcp**.

Note: This is the main page of your project's [Sphinx](#) documentation. It is formatted in [reStructuredText](#). Add additional pages by creating `.rst`-files in `docs` and adding them to the `toctree` below. Use then [references](#) in order to link them from this page, e.g. [Contributors](#) and [Changelog](#).

It is also possible to refer to the documentation of other Python packages with the [Python domain syntax](#). By default you can reference the documentation of [Sphinx](#), [Python](#), [NumPy](#), [SciPy](#), [matplotlib](#), [Pandas](#), [Scikit-Learn](#). You can add more by extending the `intersphinx_mapping` in your Sphinx's `conf.py`.

The pretty useful extension [autodoc](#) is activated by default and lets you include documentation from docstrings. Docstrings can be written in [Google style](#) (recommended!), [NumPy style](#) and [classical style](#).

1.1 License

The MIT License (MIT)

Copyright (c) 2020 nimpsch

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.

1.2 Contributors

- nimpsch <snimpsch@gmail.com>

1.3 Changelog

1.3.1 Version 0.1

- Feature A added

- FIX: nasty bug #1729 fixed
- add your changes here!

1.4 easy_tcp

1.4.1 easy_tcp package

Submodules

easy_tcp.async_client module

```
class easy_tcp.async_client.TcpClient (loop, host: str = '127.0.0.1', port: int = 8080,  
                                         auto_reconnect: bool = True)
```

Bases: `object`

Asynchronous tcp client

loop

The asyncio event loop.

Type

`obj`

host

The ip address of the tcp server.

Type `str`

port

The port of the tcp server.

Type `int`

reader

Instance of the StreamReader

Type `obj`:

writer

Instance of the StreamWriter

Type `obj`:

auto_reconnect

If true, a reconnect will be made on connection loss.

Type `bool`

logger

An instance of the logging module.

Type

`obj`

buffer

The split part of the msg which was not returned.

Type `bool`, default=True

close()

Closes the socket connection if it open

connect (*timeout: float = 10.0*)

Tries to connect to the given host. Waits 0.5 seconds until another try will be made.

Parameters **timeout** (*float, default 10.0*) – The maximum time this function will try to connect until a ClientTimeoutError is raised.

Raises ClientTimeoutError – If no connection could be established in the given time a ClientTimeoutError is raised.

is_connected

Returns True if connected.

Type bool

receive (*bytes_to_receive: int = 4096*) → bytes

Receives messages from the socket. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed, otherwise an empty byte string will be returned.

Parameters **bytes_to_receive** (*int, default 4096*) – Reads the number bytes from the socket. Returns fewer bytes than bytes_to_receive if fewer are available.

Returns The received data from the socket. Or an empty byte string if socket.error is raised.

Return type bytes

receive_until (*bytes_to_receive: int = 4096, delimiter: bytes = '\n', timeout: float = 1.0*) → bytes

Receives messages from the socket until the given delimiter is recognized.

The data will be split at the delimiter. The delimiter will be removed from the message and returned. If the received message contains a message after the delimiter, it will be stored in a buffer and prepended to the next message. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed, otherwise an empty byte string will be returned.

Parameters

- **bytes_to_receive** (*int, default 4096*) – Reads the number bytes from the socket. Returns fewer bytes than bytes_to_receive if fewer are available.
- **delimiter** (*bytes, default '\n'*) – Splits the read data at the delimiter
- **timeout** (*float, default 1.0*) – The maximum time this function will wait until a ClientTimeoutError is raised.

Returns The received data from the socket. Or an empty byte string if socket.error is raised.

Return type bytes

Raises ClientTimeoutError – Raises if no data was read or no delimiter was found withing the given time.

send (*data: bytes*)

Sends a message to the socket. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed.

Parameters **data** (*bytes*) – Sends the given bytes to the socket.

easy_tcp.client module

class easy_tcp.client.TcpClient (*host: str = '127.0.0.1', port: int = 8080, auto_reconnect: bool = True*)

Bases: object

A tcp client

host

The ip address of the tcp server.

Type `str`

port

The port of the tcp server.

Type `int`

auto_reconnect

If true, a reconnect will be made on connection loss.

Type `bool`

logger

An instance of the logging module.

Type

`obj`

buffer

The split part of the msg which was not returned.

Type `bool`, default=True

close()

Closes the socket connection if it open

connect (*timeout: float = 10.0*)

Tries to connect to the given host. Waits 0.5 seconds until another try will be made.

Parameters **timeout** (*float, default 10.0*) – The maximum time this function will try to connect until a ClientTimeoutError is raised.

Raises `ClientTimeoutError` – If no connection could be established in the given time a ClientTimeoutError is raised.

is_connected

Returns True if connected.

Type `bool`

receive (*bytes_to_receive: int = 4096*) → bytes

Receives messages from the socket. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed, otherwise an empty byte string will be returned.

Parameters **bytes_to_receive** (*int, default 4096*) – Reads the number bytes from the socket. Returns fewer bytes than bytes_to_receive if fewer are available.

Returns The received data from the socket. Or an empty byte string if socket.error is raised.

Return type `bytes`

receive_until (*bytes_to_receive: int = 4096, delimiter: bytes = '\n', timeout: float = 1.0*) → bytes

Receives messages from the socket until the given delimiter is recognized.

The data will be split at the delimiter. The delimiter will be removed from the message and returned. If the received message contains a message after the delimiter, it will be stored in a buffer and prepended to the next message. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed, otherwise an empty byte string will be returned.

bytes_to_receive [int, default 4096] Reads the number bytes from the socket. Returns fewer bytes than bytes_to_receive if fewer are available.

delimiter [bytes, default '\n'] Splits the read data at the delimiter

timeout [float, default 1.0] The maximum time this function will wait until a ClientTimeoutError is raised.

bytes The received data from the socket. Or an empty byte string if socket.error is raised.

ClientTimeoutError Raises if no data was read or no delimiter was found withing the given time.

send (data: bytes)

Send a message to the socket. If an socket.error is raised and auto_connect is enabled, a reconnect will be executed.

Parameters **data** (bytes) – Sends the given bytes to the socket.

easy_tcp.client_errors module

exception easy_tcp.client_errors.ClientError

Bases: Exception

exception easy_tcp.client_errors.ClientProtocolError

Bases: easy_tcp.client_errors.ClientError

exception easy_tcp.client_errors.ClientSocketError

Bases: easy_tcp.client_errors.ClientError

exception easy_tcp.client_errors.ClientTimeoutError

Bases: easy_tcp.client_errors.ClientError

Module contents

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

e

- `easy_tcp`, [7](#)
- `easy_tcp.async_client`, [4](#)
- `easy_tcp.client`, [5](#)
- `easy_tcp.client_errors`, [7](#)

A

auto_reconnect (*easy_tcp.async_client.TcpClient attribute*), 4
auto_reconnect (*easy_tcp.client.TcpClient attribute*), 6

B

buffer (*easy_tcp.async_client.TcpClient attribute*), 4
buffer (*easy_tcp.client.TcpClient attribute*), 6

C

ClientError, 7
ClientProtocolError, 7
ClientSocketError, 7
ClientTimeoutError, 7
close() (*easy_tcp.async_client.TcpClient method*), 4
close() (*easy_tcp.client.TcpClient method*), 6
connect() (*easy_tcp.async_client.TcpClient method*), 5
connect() (*easy_tcp.client.TcpClient method*), 6

E

easy_tcp (*module*), 7
easy_tcp.async_client (*module*), 4
easy_tcp.client (*module*), 5
easy_tcp.client_errors (*module*), 7

H

host (*easy_tcp.async_client.TcpClient attribute*), 4
host (*easy_tcp.client.TcpClient attribute*), 6

I

is_connected (*easy_tcp.async_client.TcpClient attribute*), 5
is_connected (*easy_tcp.client.TcpClient attribute*), 6

L

logger (*easy_tcp.async_client.TcpClient attribute*), 4
logger (*easy_tcp.client.TcpClient attribute*), 6

loop (*easy_tcp.async_client.TcpClient attribute*), 4

P

port (*easy_tcp.async_client.TcpClient attribute*), 4
port (*easy_tcp.client.TcpClient attribute*), 6

R

reader (*easy_tcp.async_client.TcpClient attribute*), 4
receive() (*easy_tcp.async_client.TcpClient method*), 5
receive() (*easy_tcp.client.TcpClient method*), 6
receive_until() (*easy_tcp.async_client.TcpClient method*), 5
receive_until() (*easy_tcp.client.TcpClient method*), 6

S

send() (*easy_tcp.async_client.TcpClient method*), 5
send() (*easy_tcp.client.TcpClient method*), 7

T

TcpClient (*class in easy_tcp.async_client*), 4
TcpClient (*class in easy_tcp.client*), 5

W

writer (*easy_tcp.async_client.TcpClient attribute*), 4