# easy$_{t}cpDocumentation$

***Release unknown***

**nimpsch**

**Mar 08, 2020**

# Contents

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.

# CHAPTER 1

# Contents

## 1.1 License

## 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*)
: 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 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

# Python Module Index

# Index