

SLCP-Chat

Generated by Doxygen 1.14.0

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 common Namespace Reference	7
4.1.1 Detailed Description	7
4.1.2 Function Documentation	8
4.1.2.1 create_fifo()	8
4.1.2.2 graceful_shutdown()	8
4.1.2.3 load_config()	8
4.1.2.4 pipe_path()	8
4.1.2.5 read_from_fifo()	9
4.1.2.6 remove_pid()	9
4.1.2.7 write_pid()	9
4.1.2.8 write_to_fifo()	9
4.1.3 Variable Documentation	10
4.1.3.1 exist_ok	10
4.1.3.2 format	10
4.1.3.3 INFO	10
4.1.3.4 level	10
4.1.3.5 log	10
4.1.3.6 PID_FILE	10
4.1.3.7 PIPE_DIR	10
4.1.3.8 USER	10
4.2 config Namespace Reference	10
4.2.1 Variable Documentation	11
4.2.1.1 autoreply	11
4.2.1.2 handle	11
4.2.1.3 imagepath	11
4.2.1.4 port	11
4.2.1.5 whoisport	11
4.3 discovery Namespace Reference	11
4.3.1 Detailed Description	11
4.3.2 Variable Documentation	12
4.3.2.1 d	12
4.4 network Namespace Reference	12
4.4.1 Detailed Description	12

4.4.2 Function Documentation	13
4.4.2.1 cleanup_and_exit()	13
4.4.2.2 get_own_ip()	13
4.4.2.3 handle_commands()	13
4.4.2.4 listen_tcp()	14
4.4.2.5 receive_udp()	14
4.4.2.6 send_img()	14
4.4.2.7 send_join()	14
4.4.2.8 send_leave()	15
4.4.2.9 send_msg()	15
4.4.2.10 send_who()	15
4.4.2.11 send_whois()	15
4.4.2.12 start_network()	15
4.4.3 Variable Documentation	16
4.4.3.1 config	16
4.4.3.2 FIFO_NET_TO_UI	16
4.4.3.3 FIFO_UI_TO_NET	16
4.4.3.4 last_peers_display	16
4.4.3.5 left_peers	16
4.4.3.6 peer_join_time	16
4.4.3.7 peers	16
4.4.3.8 running	16
4.4.3.9 udp_sock	16
4.5 peer Namespace Reference	17
4.5.1 Detailed Description	17
4.6 ui Namespace Reference	17
4.6.1 Detailed Description	17
4.6.2 Variable Documentation	17
4.6.2.1 FIFO_NET_TO_UI	17
4.6.2.2 FIFO_UI_TO_NET	17
5 Class Documentation	19
5.1 ui.ChatUI Class Reference	19
5.1.1 Detailed Description	19
5.1.2 Constructor & Destructor Documentation	19
5.1.2.1 __init__()	19
5.1.3 Member Function Documentation	20
5.1.3.1 listen_pipes()	20
5.1.3.2 start()	20
5.1.4 Member Data Documentation	20
5.1.4.1 config	20
5.1.4.2 handle	20

5.1.4.3 pipe_path	20
5.1.4.4 port	20
5.2 discovery.Discovery Class Reference	21
5.2.1 Detailed Description	21
5.2.2 Constructor & Destructor Documentation	21
5.2.2.1 __init__()	21
5.2.3 Member Function Documentation	21
5.2.3.1 listen_udp_all()	21
5.2.3.2 send_who()	22
5.2.3.3 shutdown()	22
5.2.3.4 start()	22
5.2.4 Member Data Documentation	22
5.2.4.1 config	22
5.2.4.2 handle	22
5.2.4.3 known_peers	22
5.2.4.4 port	22
5.2.4.5 udp_listener_thread	23
5.2.4.6 whois_port	23
5.3 peer.Peer Class Reference	23
5.3.1 Detailed Description	23
5.3.2 Constructor & Destructor Documentation	24
5.3.2.1 __init__()	24
5.3.3 Member Function Documentation	24
5.3.3.1 __eq__()	24
5.3.3.2 __hash__()	24
5.3.3.3 __repr__()	24
5.3.3.4 get_address()	25
5.3.4 Member Data Documentation	25
5.3.4.1 handle	25
5.3.4.2 ip	25
5.3.4.3 port	25
6 File Documentation	27
6.1 common.py File Reference	27
6.2 config.toml File Reference	27
6.3 discovery.py File Reference	28
6.4 network.py File Reference	28
6.5 peer.py File Reference	29
6.6 ui.py File Reference	29
Index	31

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

common	7
config	10
discovery	11
network	12
peer	17
ui	17

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ui.ChatUI	19
discovery.Discovery	21
peer.Peer	23

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

common.py	27
config.toml	27
discovery.py	28
network.py	28
peer.py	29
ui.py	29

Chapter 4

Namespace Documentation

4.1 common Namespace Reference

Functions

- `load_config` (config_path="config.toml")
- `pipe_path` (name)
- `create_fifo` (name)
- `write_to_fifo` (name, message)
- `read_from_fifo` (name, blocking=True)
- `write_pid` (pid_file=PID_FILE)
- `remove_pid` (pid_file=PID_FILE)
- `graceful_shutdown` (sig, frame)

Variables

- `USER` = `getpass.getuser()`
- `str PIPE_DIR` = `f"/tmp/slcp_{USER}"`
- `exist_ok`
- `PID_FILE` = `os.path.join(PIPE_DIR, "slcp_discovery.pid")`
- `level`
- `INFO`
- `format`
- `log` = `logging.getLogger("SLCP")`

4.1.1 Detailed Description

```
@file common.py
@brief Utility-Funktionen und IPC-/Logging-Infrastruktur für den SLCP-Discovery-Dienst.

Diese Datei stellt Hilfsfunktionen, globale Konstanten und Pipe-Verwaltung bereit.
```

4.1.2 Function Documentation

4.1.2.1 create_fifo()

```
common.create_fifo (  
    name)
```

@brief Erstellt eine Named Pipe (FIFO), falls sie nicht existiert.

@param name Name der Pipe, die erstellt werden soll.

@return Pfad zur erstellten oder bereits existierenden Pipe.

4.1.2.2 graceful_shutdown()

```
common.graceful_shutdown (  
    sig,  
    frame)
```

@brief Signal-Handler für SIGINT, beendet das Programm sauber.

@param sig Signalnummer (z.B. signal.SIGINT).

@param frame Aktueller Stack-Frame (wird nicht genutzt).

@return None

4.1.2.3 load_config()

```
common.load_config (  
    config_path = "config.toml")
```

@brief Liest die Konfiguration aus einer TOML-Datei.

Diese Funktion lädt die Datei 'config.toml' und gibt den Abschnitt 'DEFAULT' zurück.

@param config_path Pfad zur TOML-Konfigurationsdatei.

@return Dictionary mit Werten aus dem Abschnitt 'DEFAULT'.

@throws SystemExit Wenn die Datei nicht existiert oder nicht gelesen werden kann.

4.1.2.4 pipe_path()

```
common.pipe_path (  
    name)
```

@brief Bestimmt den Pfad zur FIFO für einen gegebenen Prozessnamen.

@param name Name der Pipe (z.B. 'ui', 'network').

@return Vollständiger Pfad zur Named Pipe im PIPE_DIR.

4.1.2.5 read_from_fifo()

```
common.read_from_fifo (  
    name,  
    blocking = True)
```

@brief Liest eine Zeile aus einer Named Pipe.

@param name Name der Pipe.

@param blocking Wenn False, wird non-blocking gelesen.

@return Geladene Zeile als String (ohne Newline) oder leer bei Fehler.

4.1.2.6 remove_pid()

```
common.remove_pid (  
    pid_file = PID_FILE)
```

@brief Entfernt die PID-Datei des Discovery-Dienstes.

@param pid_file Pfad zur zu entfernenden PID-Datei.

@return None

4.1.2.7 write_pid()

```
common.write_pid (  
    pid_file = PID_FILE)
```

@brief Schreibt die aktuelle Prozess-ID in eine Datei.

@param pid_file Pfad zur PID-Datei.

@return None

4.1.2.8 write_to_fifo()

```
common.write_to_fifo (  
    name,  
    message)
```

@brief Schreibt eine Nachricht in eine Named Pipe.

Versucht bis zu 5 Mal, non-blocking in die FIFO zu schreiben, und ignoriert Broken-Pipe-Fehler oder fehlende Leser.

@param name Name der Pipe.

@param message Zu schreibende Nachricht (String).

@return None

4.1.3 Variable Documentation

4.1.3.1 exist_ok

```
common.exist_ok
```

4.1.3.2 format

```
common.format
```

4.1.3.3 INFO

```
common.INFO
```

4.1.3.4 level

```
common.level
```

4.1.3.5 log

```
common.log = logging.getLogger("SLCP")
```

4.1.3.6 PID_FILE

```
common.PID_FILE = os.path.join(PIPE_DIR, "slcp_discovery.pid")
```

4.1.3.7 PIPE_DIR

```
common.PIPE_DIR = f"/tmp/slcp_{USER}"
```

4.1.3.8 USER

```
common.USER = getpass.getuser()
```

4.2 config Namespace Reference

Variables

- str `handle` = "Alice"
- int `port` = 5010
- int `whoisport` = 4000
- str `autoreply` = "Ich bin gerade nicht da."
- str `imagepath` = "images"

4.2.1 Variable Documentation

4.2.1.1 autoreply

```
str config.autoreply = "Ich bin gerade nicht da."
```

4.2.1.2 handle

```
str config.handle = "Alice"
```

4.2.1.3 imagepath

```
str config.imagepath = "images"
```

4.2.1.4 port

```
int config.port = 5010
```

4.2.1.5 whoisport

```
int config.whoisport = 4000
```

4.3 discovery Namespace Reference

Classes

- class [Discovery](#)

Variables

- `d` = [Discovery\(\)](#)

4.3.1 Detailed Description

```
@file discovery.py
```

```
@brief Implementierung des Discovery-Dienstes für das SLCP-Chat-Programm.
```

Die Klasse `Discovery` verwaltet eingehende `WHO-`, `JOIN-` und `LEAVE-`Anfragen via UDP. Sie speichert bekannte Peers und antwortet auf Broadcasts mit aktuellen Benutzer-Informationen.

4.3.2 Variable Documentation

4.3.2.1 d

```
discovery.d = Discovery()
```

4.4 network Namespace Reference

Functions

- [get_own_ip](#) (peer_ip="8.8.8.8")
- [send_join](#) (handle, port)
- [send_leave](#) (handle)
- [send_who](#) ()
- [send_whois](#) (target_handle)
- [send_msg](#) (target, text)
- [send_img](#) (target, image_path)
- [receive_udp](#) ()
- [listen_tcp](#) (port, imagepath)
- [handle_commands](#) (handle, port, imagepath)
- [cleanup_and_exit](#) (signum, frame)
- [start_network](#) ()

Variables

- str [FIFO_UI_TO_NET](#) = "ui_to_net"
- str [FIFO_NET_TO_UI](#) = "net_to_ui"
- dict [peers](#) = {}
- [udp_sock](#) = None
- dict [config](#) = {}
- bool [running](#) = True
- [last_peers_display](#) = set()
- [left_peers](#) = set()
- dict [peer_join_time](#) = {}

4.4.1 Detailed Description

```
@file network.py
@brief Implementierung der Netzwerk-Kommunikation für das SLCP-Chat-Programm.
```

Dieses Modul kümmert sich um das Senden und Empfangen von SLCP-Nachrichten:

- Broadcast (JOIN, LEAVE, WHO)
- Unicast (MSG, IMG, WHOIS, KNOWNUSERS)

Es verwendet UDP für Broadcasts und TCP für Peer-to-Peer-Kommunikation.

4.4.2 Function Documentation

4.4.2.1 cleanup_and_exit()

```
network.cleanup_and_exit (  
    signum,  
    frame)
```

@brief Signal-Handler für SIGINT/SIGTERM, sendet LEAVE und beendet das Programm.

@param *signum* Signalnummer.

@param *frame* Aktueller Stack-Frame.

@return None

4.4.2.2 get_own_ip()

```
network.get_own_ip (  
    peer_ip = "8.8.8.8")
```

@brief Ermittelt die eigene lokale IP-Adresse.

Baut eine temporäre UDP-Verbindung zu '*peer_ip*' auf und liest die lokale Socket-Adresse aus.

@param *peer_ip* Externe IP-Adresse zum Herstellen der Verbindung.

@return String mit der lokalen IP-Adresse oder '127.0.0.1' bei Fehler.

4.4.2.3 handle_commands()

```
network.handle_commands (  
    handle,  
    port,  
    imagepath)
```

@brief Verarbeitet Befehle von der UI-FIFO und ruft entsprechende Funktionen auf.

Unterstützte Befehle:

- msg <target> <text>
- img <target> <path>
- who, whois <handle>
- JOIN <handle> <port>
- LEAVE

@param *handle* Eigenes Handle.

@param *port* Eigener Port (UDP/TCP).

@param *imagepath* Verzeichnis für Bilder.

@return None

4.4.2.4 listen_tcp()

```
network.listen_tcp (  
    port,  
    imagepath)
```

@brief TCP-Server-Loop für Unicast-Kommunikation (MSG, IMG, WHOIS, JOIN, LEAVE, WHO).

@param port Eigener TCP-Port (gleich UDP-Port).
@param imagepath Verzeichnis zum Speichern empfangener Bilder.
@return None

4.4.2.5 receive_udp()

```
network.receive_udp ()
```

@brief Listener-Loop für eingehende UDP-Nachrichten (JOIN/LEAVE/IAM/KNOWNUSERS).

Schreibt entsprechende Events in die UI-FIFO und aktualisiert Peers-Listen.

@return None

4.4.2.6 send_img()

```
network.send_img (  
    target,  
    image_path)
```

@brief Sendet eine Bildnachricht (IMG) an einen Peer.

@param target Handle des Empfängers.
@param image_path Pfad zur Bilddatei.
@return None

4.4.2.7 send_join()

```
network.send_join (  
    handle,  
    port)
```

@brief Sendet eine JOIN-Nachricht per UDP-Broadcast und TCP-Unicast.

Verteilt die Nachricht an Broadcast und an bereits bekannte Peers.

@param handle Eigenes Handle.
@param port Eigener UDP-Port.
@return None

4.4.2.8 send_leave()

```
network.send_leave (  
    handle)
```

@brief Sendet eine LEAVE-Nachricht an alle Teilnehmer.

@param *handle* Eigenes Handle.
@return None

4.4.2.9 send_msg()

```
network.send_msg (  
    target,  
    text)
```

@brief Sendet eine Textnachricht (MSG) an einen Peer.

@param *target* Handle des Empfängers.
@param *text* Nachrichtentext.
@return None

4.4.2.10 send_who()

```
network.send_who ()
```

@brief Sendet eine WHO-Broadcast-Anfrage.

@return None

4.4.2.11 send_whois()

```
network.send_whois (  
    target_handle)
```

@brief Sendet eine WHOIS-Anfrage an einen spezifischen Peer.

@param *target_handle* Handle des Ziels.
@return None

4.4.2.12 start_network()

```
network.start_network ()
```

@brief Initialisiert das Netzwerk-Modul und startet Listener-Threads.

Lädt Konfiguration, baut Sockets, startet Broadcast/Unicast-Listeners und verarbeitet UI-Befehle.

@return None

4.4.3 Variable Documentation

4.4.3.1 config

```
dict network.config = {}
```

4.4.3.2 FIFO_NET_TO_UI

```
str network.FIFO_NET_TO_UI = "net_to_ui"
```

4.4.3.3 FIFO_UI_TO_NET

```
str network.FIFO_UI_TO_NET = "ui_to_net"
```

4.4.3.4 last_peers_display

```
network.last_peers_display = set()
```

4.4.3.5 left_peers

```
network.left_peers = set()
```

4.4.3.6 peer_join_time

```
dict network.peer_join_time = {}
```

4.4.3.7 peers

```
dict network.peers = {}
```

4.4.3.8 running

```
bool network.running = True
```

4.4.3.9 udp_sock

```
network.udp_sock = None
```

4.5 peer Namespace Reference

Classes

- class [Peer](#)

4.5.1 Detailed Description

```
@file peer.py
@brief Definiert die Peer-Klasse, die einen Chat-Teilnehmer modelliert.

Die Klasse Peer kapselt Handle, IP-Adresse und Port eines Chat-Teilnehmers
und stellt Methoden für Vergleich und Adresszugriff bereit.
```

4.6 ui Namespace Reference

Classes

- class [ChatUI](#)

Variables

- str [FIFO_UI_TO_NET](#) = "ui_to_net"
- str [FIFO_NET_TO_UI](#) = "net_to_ui"

4.6.1 Detailed Description

```
@file ui.py
@brief Implementierung der Kommandozeilen-Benutzeroberfläche (CLI) für das SLCP-Chat-Programm.

Die Klasse ChatUI bietet eine textbasierte Oberfläche zum Senden von Befehlen
und Anzeigen von Nachrichten/Broadcasts. Kommunikation mit dem Netzwerk-Modul erfolgt
über Named Pipes (FIFOs).
```

4.6.2 Variable Documentation

4.6.2.1 FIFO_NET_TO_UI

```
str ui.FIFO_NET_TO_UI = "net_to_ui"
```

4.6.2.2 FIFO_UI_TO_NET

```
str ui.FIFO_UI_TO_NET = "ui_to_net"
```


Chapter 5

Class Documentation

5.1 ui.ChatUI Class Reference

Public Member Functions

- `__init__` (self)
- `listen_pipes` (self)
- `start` (self)

Public Attributes

- `config` = `load_config()`
- `handle` = `self.config["handle"]`
- `port` = `self.config["port"]`
- `pipe_path` = `pipe_path(FIFO_NET_TO_UI)`

5.1.1 Detailed Description

@brief Chat-Interface für den Benutzer.

Initialisiert UI-Pipes, lädt Konfiguration und bietet eine Eingabe-Loop für Befehle.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `__init__()`

```
ui.ChatUI.__init__ (  
    self)
```

@brief Initialisiert das ChatUI-Objekt.

Lädt Konfiguration, legt Named Pipes an und speichert Pfad zur Eingabe-Pipe.

5.1.3 Member Function Documentation

5.1.3.1 listen_pipes()

```
ui.ChatUI.listen_pipes (  
    self)
```

@brief Liest dauerhaft Nachrichten aus der FIFO der Netzwerk-Komponente.
Öffnet die UI-Leser-Pipe und gibt neue Nachrichten in der Konsole aus.

5.1.3.2 start()

```
ui.ChatUI.start (  
    self)
```

@brief Startet die UI-Hauptschleife und den Pipe-Listener-Thread.
Zeigt Willkommensnachricht und Eingabe-Prompt an und verarbeitet Nutzerbefehle.

5.1.4 Member Data Documentation

5.1.4.1 config

```
ui.ChatUI.config = load_config()
```

5.1.4.2 handle

```
ui.ChatUI.handle = self.config["handle"]
```

5.1.4.3 pipe_path

```
ui.ChatUI.pipe_path = pipe_path(FIFO_NET_TO_UI)
```

5.1.4.4 port

```
ui.ChatUI.port = self.config["port"]
```

The documentation for this class was generated from the following file:

- [ui.py](#)

5.2 discovery.Discovery Class Reference

Public Member Functions

- `__init__` (self, config_path="config.toml")
- `start` (self)
- `send_who` (self)
- `listen_udp_all` (self)
- `shutdown` (self, *args)

Public Attributes

- `config` = load_config(config_path)
- `handle` = self.config["handle"]
- `port` = self.config["port"]
- `whois_port` = self.config.get("whoisport", 4000)
- dict `known_peers` = {}
- `udp_listener_thread`

5.2.1 Detailed Description

Discovery-Dienst-Klasse

Startet einen Thread, der auf alle relevanten SLCP-Befehle via UDP lauscht.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
discovery.Discovery.__init__ (
    self,
    config_path = "config.toml")
```

5.2.3 Member Function Documentation

5.2.3.1 listen_udp_all()

```
discovery.Discovery.listen_udp_all (
    self)
```

Lauscht auf UDP-Nachrichten aller SLCP-Befehle:

- WHO: Antwortet mit aktueller Peerliste.
- JOIN: Fügt neuen Peer hinzu und leitet JOIN an andere weiter.
- LEAVE: Entfernt Peer und leitet LEAVE an andere weiter.

@return None

5.2.3.2 send_who()

```
discovery.Discovery.send_who (  
    self)
```

Sendet eine WHO-Broadcast-Anfrage an alle Discovery-Server.

5.2.3.3 shutdown()

```
discovery.Discovery.shutdown (  
    self,  
    * args)
```

Beendet den Discovery-Dienst und entfernt die PID-Datei.

5.2.3.4 start()

```
discovery.Discovery.start (  
    self)
```

Startet den Discovery-Listener-Thread und hält den Dienst aktiv.

5.2.4 Member Data Documentation

5.2.4.1 config

```
discovery.Discovery.config = load_config(config_path)
```

5.2.4.2 handle

```
discovery.Discovery.handle = self.config["handle"]
```

5.2.4.3 known_peers

```
dict discovery.Discovery.known_peers = {}
```

5.2.4.4 port

```
discovery.Discovery.port = self.config["port"]
```

5.2.4.5 udp_listener_thread

```
discovery.Discovery.udp_listener_thread
```

Initial value:

```
= threading.Thread(  
    target=self.listen_udp_all, daemon=True  
)
```

5.2.4.6 whois_port

```
discovery.Discovery.whois_port = self.config.get("whoisport", 4000)
```

The documentation for this class was generated from the following file:

- [discovery.py](#)

5.3 peer.Peer Class Reference

Public Member Functions

- [__init__](#) (self, [handle](#), [ip](#), [port](#))
- [__repr__](#) (self)
- [__eq__](#) (self, other)
- [__hash__](#) (self)
- [get_address](#) (self)

Public Attributes

- [handle](#) = handle
- [ip](#) = ip
- [port](#) = port

5.3.1 Detailed Description

@brief Repräsentiert einen Chat-Teilnehmer (Peer) im SLCP-Netzwerk.

Ein Peer enthält:

- handle: Der eindeutige Benutzername des Teilnehmers
- ip: Die IP-Adresse des Teilnehmers
- port: Der Port, über den der Teilnehmer erreichbar ist

5.3.2 Constructor & Destructor Documentation

5.3.2.1 `__init__()`

```
peer.Peer.__init__ (  
    self,  
    handle,  
    ip,  
    port)
```

@brief Initialisiert einen Peer.

@param *handle* Eindeutiger Benutzername des Peers.

@param *ip* IP-Adresse des Peers.

@param *port* Portnummer des Peers.

5.3.3 Member Function Documentation

5.3.3.1 `__eq__()`

```
peer.Peer.__eq__ (  
    self,  
    other)
```

@brief Vergleich zweier Peer-Objekte nach ihrem Handle.

@param *other* Ein weiteres Objekt.

@return True, wenn 'other' ein Peer ist und denselben Handle hat.

5.3.3.2 `__hash__()`

```
peer.Peer.__hash__ (  
    self)
```

@brief Berechnet den Hash-Wert eines Peers basierend auf dem Handle.

@return Integer-Hash des Handles.

5.3.3.3 `__repr__()`

```
peer.Peer.__repr__ (  
    self)
```

@brief String-Repräsentation eines Peers.

@return Darstellung in der Form 'handle@ip:port'.

5.3.3.4 get_address()

```
peer.Peer.get_address (  
    self)
```

@brief Liefert ein Tuple aus IP-Adresse und Port für Netzwerkverbindungen.

@return Tuple (ip, port).

5.3.4 Member Data Documentation

5.3.4.1 handle

```
peer.Peer.handle = handle
```

5.3.4.2 ip

```
peer.Peer.ip = ip
```

5.3.4.3 port

```
peer.Peer.port = port
```

The documentation for this class was generated from the following file:

- [peer.py](#)

Chapter 6

File Documentation

6.1 common.py File Reference

Namespaces

- namespace [common](#)

Functions

- [common.load_config](#) (config_path="config.toml")
- [common.pipe_path](#) (name)
- [common.create_fifo](#) (name)
- [common.write_to_fifo](#) (name, message)
- [common.read_from_fifo](#) (name, blocking=True)
- [common.write_pid](#) (pid_file=[PID_FILE](#))
- [common.remove_pid](#) (pid_file=[PID_FILE](#))
- [common.graceful_shutdown](#) (sig, frame)

Variables

- [common.USER](#) = `getpass.getuser()`
- str [common.PIPE_DIR](#) = `f"/tmp/slcp_{USER}"`
- [common.exist_ok](#)
- [common.PID_FILE](#) = `os.path.join(PIPE_DIR, "slcp_discovery.pid")`
- [common.level](#)
- [common.INFO](#)
- [common.format](#)
- [common.log](#) = `logging.getLogger("SLCP")`

6.2 config.toml File Reference

Namespaces

- namespace [config](#)

Variables

- str `config.handle` = "Alice"
- int `config.port` = 5010
- int `config.whoisport` = 4000
- str `config.autoreply` = "Ich bin gerade nicht da."
- str `config.imagepath` = "images"

6.3 discovery.py File Reference

Classes

- class `discovery.Discovery`

Namespaces

- namespace `discovery`

Variables

- `discovery.d` = `Discovery()`

6.4 network.py File Reference

Namespaces

- namespace `network`

Functions

- `network.get_own_ip` (peer_ip="8.8.8.8")
- `network.send_join` (handle, port)
- `network.send_leave` (handle)
- `network.send_who` ()
- `network.send_whois` (target_handle)
- `network.send_msg` (target, text)
- `network.send_img` (target, image_path)
- `network.receive_udp` ()
- `network.listen_tcp` (port, imagepath)
- `network.handle_commands` (handle, port, imagepath)
- `network.cleanup_and_exit` (signum, frame)
- `network.start_network` ()

Variables

- str `network.FIFO_UI_TO_NET` = "ui_to_net"
- str `network.FIFO_NET_TO_UI` = "net_to_ui"
- dict `network.peers` = {}
- `network.udp_sock` = None
- dict `network.config` = {}
- bool `network.running` = True
- `network.last_peers_display` = set()
- `network.left_peers` = set()
- dict `network.peer_join_time` = {}

6.5 peer.py File Reference

Classes

- class `peer.Peer`

Namespaces

- namespace `peer`

6.6 ui.py File Reference

Classes

- class `ui.ChatUI`

Namespaces

- namespace `ui`

Variables

- str `ui.FIFO_UI_TO_NET` = "ui_to_net"
- str `ui.FIFO_NET_TO_UI` = "net_to_ui"

Index

- `__eq__`
 - `peer.Peer`, [24](#)
 - `__hash__`
 - `peer.Peer`, [24](#)
 - `__init__`
 - `discovery.Discovery`, [21](#)
 - `peer.Peer`, [24](#)
 - `ui.ChatUI`, [19](#)
 - `__repr__`
 - `peer.Peer`, [24](#)
- `autoreply`
 - `config`, [11](#)
- `cleanup_and_exit`
 - `network`, [13](#)
- `common`, [7](#)
 - `create_fifo`, [8](#)
 - `exist_ok`, [10](#)
 - `format`, [10](#)
 - `graceful_shutdown`, [8](#)
 - `INFO`, [10](#)
 - `level`, [10](#)
 - `load_config`, [8](#)
 - `log`, [10](#)
 - `PID_FILE`, [10](#)
 - `PIPE_DIR`, [10](#)
 - `pipe_path`, [8](#)
 - `read_from_fifo`, [8](#)
 - `remove_pid`, [9](#)
 - `USER`, [10](#)
 - `write_pid`, [9](#)
 - `write_to_fifo`, [9](#)
- `common.py`, [27](#)
- `config`, [10](#)
 - `autoreply`, [11](#)
 - `discovery.Discovery`, [22](#)
 - `handle`, [11](#)
 - `imagepath`, [11](#)
 - `network`, [16](#)
 - `port`, [11](#)
 - `ui.ChatUI`, [20](#)
 - `whoisport`, [11](#)
- `config.toml`, [27](#)
- `create_fifo`
 - `common`, [8](#)
- `d`
 - `discovery`, [12](#)
- `discovery`, [11](#)
 - `d`, [12](#)
 - `discovery.Discovery`, [21](#)
 - `__init__`, [21](#)
 - `config`, [22](#)
 - `handle`, [22](#)
 - `known_peers`, [22](#)
 - `listen_udp_all`, [21](#)
 - `port`, [22](#)
 - `send_who`, [21](#)
 - `shutdown`, [22](#)
 - `start`, [22](#)
 - `udp_listener_thread`, [22](#)
 - `whois_port`, [23](#)
 - `discovery.py`, [28](#)
- `exist_ok`
 - `common`, [10](#)
- `FIFO_NET_TO_UI`
 - `network`, [16](#)
 - `ui`, [17](#)
- `FIFO_UI_TO_NET`
 - `network`, [16](#)
 - `ui`, [17](#)
- `format`
 - `common`, [10](#)
- `get_address`
 - `peer.Peer`, [24](#)
- `get_own_ip`
 - `network`, [13](#)
- `graceful_shutdown`
 - `common`, [8](#)
- `handle`
 - `config`, [11](#)
 - `discovery.Discovery`, [22](#)
 - `peer.Peer`, [25](#)
 - `ui.ChatUI`, [20](#)
- `handle_commands`
 - `network`, [13](#)
- `imagepath`
 - `config`, [11](#)
- `INFO`
 - `common`, [10](#)
- `ip`
 - `peer.Peer`, [25](#)
- `known_peers`
 - `discovery.Discovery`, [22](#)

- last_peers_display
 - network, 16
- left_peers
 - network, 16
- level
 - common, 10
- listen_pipes
 - ui.ChatUI, 20
- listen_tcp
 - network, 13
- listen_udp_all
 - discovery.Discovery, 21
- load_config
 - common, 8
- log
 - common, 10
- network, 12
 - cleanup_and_exit, 13
 - config, 16
 - FIFO_NET_TO_UI, 16
 - FIFO_UI_TO_NET, 16
 - get_own_ip, 13
 - handle_commands, 13
 - last_peers_display, 16
 - left_peers, 16
 - listen_tcp, 13
 - peer_join_time, 16
 - peers, 16
 - receive_udp, 14
 - running, 16
 - send_img, 14
 - send_join, 14
 - send_leave, 14
 - send_msg, 15
 - send_who, 15
 - send_whois, 15
 - start_network, 15
 - udp_sock, 16
- network.py, 28
- peer, 17
- peer.Peer, 23
 - __eq__, 24
 - __hash__, 24
 - __init__, 24
 - __repr__, 24
 - get_address, 24
 - handle, 25
 - ip, 25
 - port, 25
- peer.py, 29
- peer_join_time
 - network, 16
- peers
 - network, 16
- PID_FILE
 - common, 10
- PIPE_DIR
 - common, 10
- pipe_path
 - common, 8
 - ui.ChatUI, 20
- port
 - config, 11
 - discovery.Discovery, 22
 - peer.Peer, 25
 - ui.ChatUI, 20
- read_from_fifo
 - common, 8
- receive_udp
 - network, 14
- remove_pid
 - common, 9
- running
 - network, 16
- send_img
 - network, 14
- send_join
 - network, 14
- send_leave
 - network, 14
- send_msg
 - network, 15
- send_who
 - discovery.Discovery, 21
 - network, 15
- send_whois
 - network, 15
- shutdown
 - discovery.Discovery, 22
- start
 - discovery.Discovery, 22
 - ui.ChatUI, 20
- start_network
 - network, 15
- udp_listener_thread
 - discovery.Discovery, 22
- udp_sock
 - network, 16
- ui, 17
 - FIFO_NET_TO_UI, 17
 - FIFO_UI_TO_NET, 17
- ui.ChatUI, 19
 - __init__, 19
 - config, 20
 - handle, 20
 - listen_pipes, 20
 - pipe_path, 20
 - port, 20
 - start, 20
- ui.py, 29
- USER
 - common, 10

- whois_port
 - discovery.Discovery, [23](#)
- whoisport
 - config, [11](#)
- write_pid
 - common, [9](#)
- write_to_fifo
 - common, [9](#)