My Project

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 config Namespace Reference

Classes

- class Config

3.1.1 Detailed Description

```
@file config.py
@brief Modul zur Verwaltung der TOML-basierten Konfiguration des Chat-Clients.
@details Dieses Modul stellt die Klasse Config bereit, die eine Konfigurationsdatei im TOML-Format lädt, analy
         Sie bietet Zugriff auf zentrale Parameter wie Benutzerhandle, Portbereich, Whois-Port, Autoreply-Text
         Zusätzlich unterstützt sie das Speichern von Konfigurationsänderungen zurück in die Datei.
```

3.2 discovery Namespace Reference

Functions

- str _get_local_ip ()
- None run_discovery_service (pipe_cmd, pipe_evt, config)

Variables

- str **BROADCAST_ADDR** = '255.255.255.255'
- int **BUFFER_SIZE** = 4096

3.2.1 Detailed Description

```
@file discovery.py
@brief Discovery-Dienst für den dezentralen Chat-Client (SLCP-Protokoll).
@details Dieses Modul implementiert einen UDP-basierten Mechanismus zur automatischen
Erkennung von Teilnehmern im lokalen Netzwerk. Mithilfe von Broadcast-Nachrichten
werden Join- und Leave-Events sowie Nutzerlisten ausgetauscht.

Der Discovery-Service ist zuständig für:
- Verwaltung einer lokalen Teilnehmer-Registry
- Entgegennahme von JOIN, LEAVE, WHO-Nachrichten über UDP
- Senden von KNOWNUSERS-Antworten an andere Clients
- Synchronisation mit der UI über IPC-Pipes

Es wird ein Hintergrund-Thread verwendet, um eingehende Nachrichten parallel zur
Steuerung durch die Benutzeroberfläche (UI) zu verarbeiten.
```

3.2.2 Function Documentation

3.2.2.1 _get_local_ip()

```
str discovery._get_local_ip ( )  [protected]
```

```
@brief Ermittelt die lokale IP-Adresse des Hosts.
@details Öffnet kurz ein UDP-Socket zu einem öffentlichen Server (hier Google DNS),
         um die tatsächliche Interface-IP abzurufen. Fallback auf 127.0.0.1 bei Fehler.
@return Lokale IP-Adresse als String.
```

3.2.2.2 run_discovery_service()

```
None discovery.run_discovery_service (
              pipe_cmd,
              pipe_evt,
              config )
```

```
@brief Startet den Discovery-Service für Peer-to-Peer-Erkennung über UDP-Broadcast.
@param pipe_cmd Pipe für Steuerbefehle von der UI (join, leave, who).
@param pipe_evt Pipe zur Rückmeldung von Nutzerlisten an die UI.
@param config Konfigurationsobjekt mit Informationen über whois-Port, Handle, usw.
@details
- Verwaltet eine interne Registry aller bekannten Peers im lokalen Netzwerk.
- Reagiert auf JOIN-/LEAVE-/WHO-Anfragen.
- Antwortet mit KNOWNUSERS-Nachrichten an andere Discovery-Instanzen.
- Nutzt einen Listener-Thread, um UDP-Messages asynchron zu empfangen.
```

3.3 discovery_main Namespace Reference

Variables

- **config** = Config(sys.argv[1])
- int **ipc_port** = int(sys.argv[2]) if len(sys.argv) > 2 else config.whoisport
- str **lockfile** = f"/tmp/chat_discovery_{ipc_port}.lock"
- **fd** = os.open(lockfile, os.O_CREAT | os.O_EXCL | os.O_RDWR)
- tuple **address** = ('localhost', ipc_port)
- **authkey**
- **conn** = listener.accept()

3.3.1 Detailed Description

```
@file discovery_main.py
@brief Startskript für den Discovery-Service im dezentralen Chat-Programm.
@details Verwendet ein Lockfile, um sicherzustellen, dass pro Port nur eine Instanz läuft.
@usage python3 discovery_main.py <config.toml> [ipc_port]
```

3.4 gui Namespace Reference

Classes

- class ChatClientGUI

Functions

- None start_gui (str config_path)

Variables

- **root** = tk.Tk(); root.withdraw()
- cfg

3.4.1 Detailed Description

```
@file gui.py
@brief Erweiterte grafische Chat-Oberfläche für dezentrale Netzwerkkommunikation mit Zusatzfunktionen.
@details Diese Anwendung implementiert eine vollständige grafische Chat-Benutzeroberfläche (GUI)
         mit Peer-to-Peer-Kommunikation unter Verwendung eines Discovery-Dienstes und eines Netzwerkdienstes.
         Neben dem Senden von Textnachrichten und Bildern unterstützt sie AFK-Modus, Broadcasting,
         Join/Leave-Logik, farbliche Zuordnung der Handle-Namen und automatischen Verbindungsaufbau.
```

3.4.2 Function Documentation

3.4.2.1 start_gui()

```
None gui.start_gui (
             str config_path )
```

```
@fn start_gui
@brief Startet die grafische Chat-Oberfläche mit gegebener Konfiguration.
@param config_path Pfad zur TOML-Konfigurationsdatei, die Nutzerinformationen und Netzwerkeinstellungen enthäl
```

3.4.3 Variable Documentation

3.4.3.1 cfg

```
gui.cfg
```

Initial value:

```
00001 =  filedialog.askopenfilename(
00002             title="Konfig auswählen",
00003             filetypes=[("TOML","*.toml")],
00004             initialdir=os.path.join(os.getcwd(),'config')
00005         )
```

3.5 main Namespace Reference

Functions

- main ()

3.5.1 Detailed Description

```
@file main.py
@brief Startskript für Discovery-, Network- und UI-Prozesse.
@details Lädt die Konfigurationsdatei, startet Discovery- und Network-Services als Hintergrundprozesse
         und führt die Benutzeroberfläche im Hauptprozess aus.
@usage python3 main.py <alice.toml|bob.toml>
```

3.5.2 Function Documentation

3.5.2.1 main()

```
main.main ( )
```

```
@brief Hauptfunktion: Initialisiert die Konfiguration und startet die drei Hauptprozesse.
@detail Lädt die TOML-Konfigurationsdatei (Handle, Port-Range, Autoreply, Image-Pfad),
        richtet bidirektionale Pipes für IPC ein, startet Discovery- und Network-Services
        als Hintergrundprozesse (Daemons) und führt die UI-Schleife im Hauptprozess aus.
```

3.6 main_ui Namespace Reference

Variables

- **config** = Config(sys.argv[1])
- int **disc_port** = int(sys.argv[2]) if len(sys.argv) > 2 else 6000
- int **net_port** = int(sys.argv[3]) if len(sys.argv) > 3 else 6001
- tuple **disc_addr** = ('localhost', disc_port)
- **disc_conn** = Client(disc_addr, authkey=b'ipc_secret')
- tuple **net_addr** = ('localhost', net_port)
- **net_conn** = Client(net_addr, authkey=b'ipc_secret')

3.6.1 Detailed Description

```
@file main_ui.py
@brief Startskript für die Kommandozeilen-Oberfläche (CLI) des Chat-Clients.
@details Stellt IPC-Verbindungen zu Discovery- und Network-Services her und startet die interaktive Benutzerob
@usage python3 main_ui.py <configfile.toml> [disc_port] [net_port]
```

3.7 network Namespace Reference

Functions

- _handle_tcp (conn, pipe_evt)
- _tcp_listener (server_socket, pipe_evt)
- _udp_listener (udp_sock, pipe_evt, image_dir)
- run_network_service (pipe_cmd, pipe_evt, config)

Variables

- int **_CHUNK_SIZE** = 60000

3.7.1 Detailed Description

```
@file network.py
@brief Netzwerkdienst: SLCP-Nachrichten (MSG, IMG) über TCP/UDP senden und empfangen.
@details Implementiert einen TCP-Server für Textnachrichten und einen UDP-Server für Bilddaten.
```

3.7.2 Function Documentation

3.7.2.1 _handle_tcp()

```
network._handle_tcp (
              conn,
              pipe_evt )  [protected]
```

```
@brief Bearbeitet eine einzelne TCP-Verbindung für SLCP-MSG.
@param conn Aktive TCP-Verbindung (Socket).
@param pipe_evt Pipe zum UI-Prozess.
```

3.7.2.2 _tcp_listener()

```
network._tcp_listener (
              server_socket,
              pipe_evt )  [protected]
```

```
@brief Akzeptiert eingehende TCP-Verbindungen und leitet sie weiter.
@param server_socket Gebundener TCP-Server-Socket.
@param pipe_evt Pipe zum UI-Prozess.
```

3.7.2.3 _udp_listener()

```
network._udp_listener (
              udp_sock,
              pipe_evt,
              image_dir )  [protected]
```

```
@brief Empfängt SLCP IMG-Nachrichten per UDP, speichert Bilder und sendet Events.
@param udp_sock Gebundener UDP-Socket für Bilddaten.
@param pipe_evt Pipe zum UI-Prozess.
@param image_dir Verzeichnis zum Speichern der Bilder.
```

3.7.2.4 run_network_service()

```
network.run_network_service (
              pipe_cmd,
              pipe_evt,
              config )
```

```
@brief Netzwerkdienst: Empfängt und sendet SLCP-Nachrichten (MSG, IMG) per TCP und UDP.
@param pipe_cmd Pipe für Befehle vom UI-Prozess.
@param pipe_evt Pipe für Ereignisse an den UI-Prozess.
@param config Konfigurationsobjekt mit port_range, handle und imagepath.
```

3.8 network_main Namespace Reference

Variables

- **config** = Config(sys.argv[1])
- int **ipc_port** = int(sys.argv[2]) if len(sys.argv) > 2 else 6001
- tuple **address** = ('localhost', ipc_port)
- **authkey**
- **conn** = listener.accept()

3.8.1 Detailed Description

```
@file network_main.py
@brief Startskript für den Network-Service-Prozess.
@details Dieses Modul wird als separates Skript aufgerufen, um den Netzwerkdienst des
Chatprogramms zu starten. Es stellt eine IPC-Verbindung zur Benutzeroberfläche her
und übergibt dann die Kontrolle an den eigentlichen Netzwerkdienst.
Wird typischerweise aufgerufen durch:
@code
python3 network_main.py <config.toml> [ipc_port]
@endcode

@author Gruppe A11
@date 2025
```

3.9 ui Namespace Reference

Functions

- run_ui (pipe_net_cmd, pipe_net_evt, pipe_disc_cmd, pipe_disc_evt, config)

Variables

- **autoreset**
- _COLOR_CYCLE

3.9.1 Detailed Description

```
@file ui.py
@brief Kommandozeilen-Oberfläche (CLI) des dezentralen Chat-Programms.
@details Dieses Modul bietet eine textbasierte Benutzerschnittstelle zur Steuerung
          des Chat-Clients. Es unterstützt das Senden von Text- und Bildnachrichten,
          Teilnehmerverwaltung via Discovery-Dienst, sowie eine manuell aktivierbare
          Autoreply-Funktion bei Abwesenheit.

Verwendet ANSI-Farben zur Anzeige und bietet Konfigurationsänderungen zur Laufzeit.
@author Gruppe A11
@date 2025
```

3.9.2 Function Documentation

3.9.2.1 run_ui()

```
ui.run_ui (
              pipe_net_cmd,
              pipe_net_evt,
              pipe_disc_cmd,
              pipe_disc_evt,
              config )
```

```
@brief Startet die Kommandozeilen-Oberfläche (UI) des Chatprogramms.
@param pipe_net_cmd Pipe zur Übermittlung von Befehlen an den Netzwerkdienst.
@param pipe_net_evt Pipe für eingehende Netzwerkereignisse (z. B. empfangene Nachrichten).
@param pipe_disc_cmd Pipe zur Steuerung des Discovery-Dienstes (JOIN, WHO, LEAVE).
@param pipe_disc_evt Pipe für Discovery-Ereignisse (z. B. neue Nutzerliste).
@param config Das geladene Konfigurationsobjekt (handle, autoreply, ports, Farben).
```

3.9.3 Variable Documentation

3.9.3.1 _COLOR_CYCLE

```
ui._COLOR_CYCLE  [protected]
```

Initial value:

```
00001 =  cycle([
00002     Fore.RED, Fore.GREEN, Fore.YELLOW,
00003     Fore.BLUE, Fore.MAGENTA, Fore.CYAN
00004 ])
```

Chapter 4

Class Documentation

4.1 gui.ChatClientGUI Class Reference

Public Member Functions

- **__init__** (self, str config_path)
- None **load_config** (self, str config_path)
- None toggle_chat_status (self)
- None toggle_afk (self)
- None on_peer_select (self, event)
- None disc_listener (self)
- None net_listener (self)
- None update_peer_list (self)
- None display_message (self, str sender, str text)
- None display_image (self, str sender, str image_path)
- None send_message (self)
- None send_broadcast_message (self)
- None send_image (self)
- None on_close (self)

Public Attributes

- **afk_mode**
- **autoreply_text**
- **chat_images**
- **config**
- **handle**
- **handle_colors**
- **net_cmd**
- **net_evt**
- **disc_cmd**
- **disc_evt**
- **disc_proc**
- **net_proc**
- **root**
- **peers**
- **peer_list**

- **on_peer_select**
- **chat_display**
- **entry_text**
- **send_btn**
- **img_btn**
- **in_chat**
- **chat_toggle_btn**
- **quit_btn**
- **selected_label**
- **broadcast_btn**
- **afk_btn**
- **stop_event**

Protected Member Functions

- None **_setup_services** (self)
- None **_build_gui** (self)
- None **_create_menu** (self)
- None **_create_widgets** (self)
- None _start_listeners (self)
- None _auto_join (self)
- None _open_config_dialog (self)

4.1.1 Detailed Description

```
@class ChatClientGUI
@brief Hauptklasse der grafischen Benutzeroberfläche (GUI) für den Chat-Client.
@details Diese Klasse initialisiert die Benutzeroberfläche, lädt die Konfiguration,
         stellt IPC-Verbindungen zu Discovery- und Netzwerkdiensten her und verarbeitet
         alle UI-Interaktionen und Benutzerkommandos. Die Klasse unterstützt AFK-Modus,
         Broadcasting, Join/Leave-Logik und Bildübertragung. Die Nachrichtenverarbeitung
         erfolgt asynchron über dedizierte Listener-Threads.
```

4.1.2 Member Function Documentation

4.1.2.1 _auto_join()

```
None gui.ChatClientGUI._auto_join (
              self )  [protected]
```

```
@brief Automatischer Beitritt zum Netzwerk beim Start.
```

4.1.2.2 _open_config_dialog()

```
None gui.ChatClientGUI._open_config_dialog (
              self )  [protected]
```

```
@brief Öffnet Dialog zur Auswahl einer neuen TOML-Konfigurationsdatei.
@details Startet die Anwendung neu mit der gewählten Konfiguration.
```

4.1.2.3 _start_listeners()

```
None gui.ChatClientGUI._start_listeners (
              self )  [protected]
```

@brief Startet Hintergrund-Threads für Netzwerk- und Discovery-Events.

4.1.2.4 disc_listener()

```
None gui.ChatClientGUI.disc_listener (
              self )
```

@brief Reagiert auf Änderungen der Discovery-Teilnehmerliste.

4.1.2.5 display_image()

```
None gui.ChatClientGUI.display_image (
              self,
             str sender,
             str image_path )
```

@brief Zeigt ein Bild im Chatfenster an.
@param sender Name des Absenders.
@param image_path Dateipfad des empfangenen Bildes.

4.1.2.6 display_message()

```
None gui.ChatClientGUI.display_message (
              self,
             str sender,
             str text )
```

@brief Zeigt eine Textnachricht im Chatfenster an.
@param sender Name des Absenders.
@param text Inhalt der Nachricht.

4.1.2.7 net_listener()

```
None gui.ChatClientGUI.net_listener (
              self )
```

@brief Verarbeitet eingehende Nachrichten, Bilder oder Netzwerkfehler.
@details Antwortet bei aktivem AFK-Modus automatisch auf eingehende Nachrichten.

4.1.2.8 on_close()

```
None gui.ChatClientGUI.on_close (
              self )
```

```
@brief Beendet die Anwendung und informiert alle verbundenen Peers über das Verlassen.
@details Sendet eine "leave"-Nachricht an den Discovery-Dienst und eine Systemmeldung
         an alle aktiven Peers. Beendet anschließend Discovery- und Netzwerkprozess.
```

4.1.2.9 on_peer_select()

```
None gui.ChatClientGUI.on_peer_select (
              self,
              event )
```

```
@brief Aktualisiert die Anzeige des ausgewählten Chatpartners in der GUI.
```

4.1.2.10 send_broadcast_message()

```
None gui.ChatClientGUI.send_broadcast_message (
              self )
```

```
@brief Sendet eine Nachricht an alle bekannten Peers.
```

4.1.2.11 send_image()

```
None gui.ChatClientGUI.send_image (
              self )
```

```
@brief Sendet ein Bild an den aktuell ausgewählten Peer.
```

4.1.2.12 send_message()

```
None gui.ChatClientGUI.send_message (
              self )
```

```
@brief Sendet eine Textnachricht an den aktuell ausgewählten Peer.
```

4.1.2.13 toggle_afk()

```
None gui.ChatClientGUI.toggle_afk (
              self )
```

```
@brief Aktiviert oder deaktiviert den Abwesenheitsmodus (AFK).
@details Deaktiviert Eingabefunktionen bei Abwesenheitsmodus.
```

4.1.2.14 toggle_chat_status()

```
None gui.ChatClientGUI.toggle_chat_status (
              self )
```

```
@brief Ermöglicht das manuelle Verlassen und Wiederbeitreten zum Chat.
@details Aktualisiert GUI und informiert andere Teilnehmer.
```

4.1.2.15 update_peer_list()

```
None gui.ChatClientGUI.update_peer_list (
              self )
```

```
@brief Aktualisiert die Peer-Anzeige in der GUI.
```

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

- gui.py

4.2 config.Config Class Reference

Public Member Functions

- __init__ (self, str path)
- None save (self)

Public Attributes

- **path**
- **handle**
- **port_range**
- **whoisport**
- **autoreply**
- **imagepath**
- **handle_colors**

Protected Member Functions

- None _load (self)

4.2.1 Detailed Description

```
@file config.py
@brief Lädt und speichert die TOML-Konfiguration für den Chat-Client.
@details Legt Attribute an: `handle`, `port_range`, `whoisport`, `autoreply`, `imagepath`, `handle_colors`.
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
config.Config.__init__ (
              self,
             str path )
```

```
@brief Konstruktor: Initialisiert Config mit Datei-Pfad und lädt Inhalte.
@param[in] path Pfad zur TOML-Konfigurationsdatei.
@raises FileNotFoundError wenn Datei nicht existiert.
```

4.2.3 Member Function Documentation

4.2.3.1 _load()

```
None config.Config._load (
              self )  [protected]
```

```
@brief Interne Methode: Liest TOML-Datei und setzt Config-Attribute.
@raises KeyError falls Pflichtfelder (`handle`, `port`, `whoisport`) fehlen.
```

4.2.3.2 save()

```
None config.Config.save (
              self )
```

```
@brief Speichert aktuelle Config-Attribute zurück in die TOML-Datei.
```

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

- config.py

Index