Development of the EOF Chat UI

Nico Schottelius (nico-hsz-t (o) schottelius.org)

May 17, 2012

Contents

List of Tables

Chapter 1

Introduction

This paper describes the work to *write a chat front end that connects to a chat server via TCP*. The chat server is usually abbreviated by *CS*, the user interface is called *UI*.

1.1 Description

The aim is to write a generic chat front end for a fictional chat server that accepts a given command set. In particular all parts of the communication are unambiguous and commands sent by either side are acknowledged from the other side.

1.2 Objectives

1. Research and define set of commands to be supported by chat server
2. Define protocol to be used between server and client
3. Define transport mechanism
4. Create prototype implementation for client and test with mock server
5. Present results in class

1.3 Expected Results

1. List of supported commands
2. Protocol definition
3. Description of the underlying transport mechanism
4. Source code including build and run instructions

5. Presentation

Chapter 2

Protocol definition

2.1 Basic data types ("'EOFbdt"')

This section specifies the **b**asic **d**ata**t**ypes. They are further referenced as "'EOFbdt"'.

2.1.1 The zero byte

The zero byte is a byte with the value 0.

2.1.2 ASCII numbers

ASCII numbers use the decimal string representation of a number. ASCII numbers are often used in a packet header. ASCII numbers are used to specify the length of the packet (excluding itself). Due to compatibility of UTF-8 and ASCII, ASCII numbers may also be referred to as *UTF-8 numbers*.

2.1.3 Strings in general

Strings are transmitted without termination (i.e. no new line, no 0 byte). The encoding to be used is **UTF-8**.

2.1.4 Fixed length strings

Fixed length strings contain exactly the specified number of bytes: A 128-byte fixed length string consists of at most 128 bytes of text. If the text it contains is shorter than the specified length, it must be padded with zero bytes.

2.1.5 Variable length strings

This protocol does not specify any variable length strings.

2.2 EOF simple data types ("'EOFsdt"')

The following sections define the simple datatypes. They are further referenced as "'EOFsdt"'.

2.2.1 Command

A command is represented as an ASCII number in a fixed length string of 4 bytes. It is used to identify the intent of a message.

Examples

- 1100
- 3000
- 2200

2.2.2 Identification string (id)

To identify a message, a message may contain an identification string, called the *EOFID*. This ID is an integer that is encoded based on the following characters:

- A-Z (alphabet in upper case)
- a-z (alphabet in lower case)
- 0-9 (the digits)
- ! (exclamation mark)
- - (minus)

The order of the characters is as follows:
{0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-!}.
The length of an EOFID is 6 bytes, which results in *68719476736* possible ids.[1]. The given characters where selected to allow easy debugging.

Examples

The following examples en- and decode integers into the specified format. Use is made of the Python reference implementation:

[1] $(10 + 26 + 26 + 2)^6$

```
>>> import ceof
>>> ceof.EOFID.int_to_id(42)
'00000G'
>>> ceof.EOFID.int_to_id(1)
'000001'
>>> ceof.EOFID.int_to_id(64)
'000010'
>>> ceof.EOFID.id_to_int('000010')
64
>>> ceof.EOFID.id_to_int('!!!!!!')
68719476735
>>> ceof.EOFID.id_to_int('a-----')
11794116542
>>> ceof.EOFID.id_to_int('000000')
0
```

2.2.3 Size (size)

A size is represented as an ASCII number in a fixed length string of 6 bytes.

Examples

```
>>> import ceof
>>> ceof.fillup("10", 6)
'10\x00\x00\x00\x00'
>>> ceof.fillup("10000", 6)
'10000\x00'
>>> ceof.fillup("100000", 6)
'100000'
```

2.2.4 Peer name (name)

The peer name is a 128 byte fixed length string.

Examples

```
>>> import ceof
>>> ceof.fillup("Thomas Hü", 128)
'Thomas Hü\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.2.5 Group name (group)

The group name is a 128 byte fixed length string.

Examples

```
>>> import ceof
>>> ceof.fillup("!eof", 128)
'!eof\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.2.6 Message text (msgtxt)

The message text is a 256 byte fixed length string.

Examples

```
>>> import ceof
>>> ceof.fillup("Hello, !eof!", 256)
'Hello, !eof!\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.2.7 Peer address (address)

The address of a peer, which is a 128 byte fixed length string. Peer addresses are specified as URLs as defined in RFC3986[1].

Examples

```
>>> import ceof
>>> ceof.fillup("tcp://127.0.0.1:6667", 128)
'tcp://127.0.0.1:6667\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.2.8 Peer fingerprint (keyid)

A (PGP) fingerprint[2] is a 40 byte fixed length string. As the fingerprint has a fix length of 40 bytes, there is never padding needed.

Examples

```
% gpg --fingerprint | grep "Key fingerprint =" | sed -e 's/.*=//' -e 's/ //g'
A35767A98CA9CC3CE368679AB679548202C9B17D
```

[2]See RFC 2440[2], 11.2. Key IDs and Fingerprints

2.3 Interface between the chat server and the user interface ("'cs2ui"')

This section specifies how the user interface (UI) communicates with the chat server (CS).

2.3.1 Connection

The chat server provides a TCP listener on port 4242, to which the UI connects to. Alternate ports may be used, but need to be specified explicitly.

2.3.2 Messages

All messages exchanged between CS and the UI are represented as a series of fixed length strings. Every message begins with an **eof command**. Messages send by the chat server use eof commands beginning with **11**, messages send by the UI use eof commands that begin with **21**.

2.3.3 Message 1100: Acknowledge

This is a general acknowledge answer. The previous request from the UI with the same *ID* was successful.

Parameters

Table 2.1: Message 1100 parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12

Example

`1100abfudh`

2.3.4 Message 1101: Failure

This is a general failure answer. The previous request from the UI with the same *ID* failed. Details are specified in the reason message.

Parameters

Table 2.2: Message 1101 parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Reason	EOFsdt: msgtxt	Specifies the failure reason	Too many UIs connected.

If the failed command was ”‘2100”’, the CS will close the socket afterwards.

Example

```
>>> import ceof
>>> ceof.fillup("1101abfudhThe Error Reason", 4+6+256)
’1101abfudhThe Error
Reason\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00’
```

2.3.5 Message 1102: Exit requested

This is a shutdown request to the UI. After this message, the CS will exit.

Parameters

Table 2.3: Message 1102 parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12

Example

```
1102abf93a
```

2.3.6 Message 1103: Received message

This message is issued by the CS if a message is received.

Parameters

Table 2.4: Message 1103 parameters

Parameter	Type	Description	Example

ID	EOFsdt: id	packet id	afdb12
name	EOFsdt: name	The sender	telmich
message	EOFsdt: msgtxt	The message	Hallo, mein Freund!

Example

```
>>> import ceof
>>> cmdid="1103abcdef"
>>> name=ceof.fillup("telmich", 128)
>>> message=ceof.fillup("Hallo, mein Freund!", 256)
>>> cmdid + name + message
'1103abcdeftelmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00Hallo, mein Freund!\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00'
```

Possible answers

- None

2.3.7 Message 1104: List of peers

This is the answer to command *2106*. It contains the same ID, as the *2106* request command.

Parameters

Table 2.5: Message 1104 parameters

Parameter	**Type**	**Description**	**Example**
ID	EOFsdt: id	packet id	afdb12

Number of peers (nop)	EOFsdt: size	How many peers follow	20
nop ∗ Peer	EOFsdt: name	The name	telmich

The last field is repeated as many times as specified in the number of peers field.

Example

```
>>> import ceof
>>> cmd="1104"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> nop=ceof.fillup("2", 6)
>>> peer1=ceof.fillup("telmich", 128)
>>> peer2=ceof.fillup("Hans-Jürgen", 128)
>>> cmd + id + nop + peer1 + peer2
'1104Y7Spet2\x00\x00\x00\x00\x00telmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00Hans-Jürgen\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.3.8 Message 1105: Peer information

This is the answer to command *2105*. It contains the same ID as the *2105* request command.

Parameters

Table 2.6: Message 1105 parameters

Parameter	**Type**	**Description**	**Example**
ID	EOFsdt: id	packet id	afdb12
Keyid	EOFsdt: keyid	This peers pgp-keyid	389E5481065EAA253...
Number of addresses (noa)	EOFsdt: size		2
noa ∗ address	EOFsdt: address	Adress of peer	tcp://127.0.0.1:4243

The last field is repeated as often as specified in the number of addresses field.

Example

```
>>> import ceof
>>> cmd="1105"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> keyid="A35767A98CA9CC3CE368679AB679548202C9B17D"
>>> noa=ceof.fillup("2", 6)
>>> addr1=ceof.fillup("tcp://10.2.2.3:4242", 128)
>>> addr2=ceof.fillup("email://nico-eof42@schottelius.org", 128)
>>> cmd + id + keyid + noa + addr1 + addr2
'1105o0mZGMA35767A98CA9CC3CE368679AB679548202C9B17D2\x00\x00\x00\x00\x00
tcp://10.2.2.3:4242\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
email://nico-eof42@schottelius.org\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00'
```

2.3.9 Message 1106: Peer renamed

This is the answer to command *2104*. It contains the same ID as the *2104* request command. It is sent out to **all** connected user interfaces.

Parameters

Table 2.7: Message 1106 parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Old peer name	EOFsdt: name	Old name	susi
New peer name	EOFsdt: name	New name	heinz

Possible answers

- None

Example

```
>>> import ceof
>>> cmd="1106"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> oldname=ceof.fillup("telmich", 128)
>>> newname=ceof.fillup("Another Name", 128)
>>> cmd + id + oldname + newname
'1106FpSVy7telmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00Another Name\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00'
```

2.3.10 Message 2100: Register user interface

This must be the *first* message sent by the UI. If the answer is not *1100*, the UI should close the connection afterwards.

Parameters

Table 2.8: Message 2100 parameters

Parameter	**Type**	**Description**	**Example**
ID	EOFsdt: id	packet id	afdb12
Name	EOFsdt: name	Name of the UI	ceofui

Possible answers

- 1100
- 1101

Example

```
>>> import ceof
>>> cmd="2100"
>>> id = ceof.EOFID.int_to_id(42)
>>> name=ceof.fillup("ceofui", 128)
>>> cmd + id + name
```

```
'210000000Gceofui\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00'
```

2.3.11 Message 2101: Deregister user interface

Parameters

- none

Possible answers

- none

The CS will close the connection to the UI after receiving this message.

Example

```
>>> import ceof
>>> cmd="2101"
>>> cmd
'2101'
```

2.3.12 Message 2102: /peer add

The UI adds a peer to the list of known peers.

Parameters

Table 2.9: Message 2102: /peer add parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Peer name	EOFsdt: name	Name you identify the peer with	telmich
Address	EOFsdt: address	Where we can make the first contact	tcp://10.0.42.42:4242
Keyid	EOFsdt: keyid	PGP fingerprint of the peers key	F27987E34E66...

Possible answers

- 1100

- 1101

Example

```
>>> import ceof
>>> cmd="2102"
>>> name=ceof.fillup("telmich", 128)
>>> address=ceof.fillup("tcp://127.0.0.1:6667", 128)
>>> keyid="A35767A98CA9CC3CE368679AB679548202C9B17D"
>>> id = ceof.EOFID.int_to_id(42)
>>> cmd + id + name + address + keyid
'210200000Gtelmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00tcp://127.0.0.1:6667\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
A35767A98CA9CC3CE368679AB679548202C9B17D'
```

2.3.13 Message 2103: /peer del

The UI wants to delete a peer.

Parameters

Table 2.10: Message 2103: /peer del parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Name	EOFsdt: name	Name you identify the peer with	telmich

Possible answers

- 1100
- 1101

Example

```
>>> import ceof
>>> cmd="2103"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> name=ceof.fillup("telmich", 128)
>>> cmd + id + name
'2103hSJceCtelmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00'
```

2.3.14 Message 2104: /peer rename

This messages is issued by the UI when it wants to rename a peer.

Parameters

Table 2.11: Message 2104: /peer rename parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Old name	EOFsdt: name	Old peer name	susi
New name	EOFsdt: name	New peer name	heinz

Possible answers

- 1106
- 1101

Example

```
>>> import ceof
>>> cmd="2104"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> oldname=ceof.fillup("telmich", 128)
>>> newname=ceof.fillup("Another Name", 128)
>>> cmd + id + oldname + newname
'2104zMpcG8telmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00Another Name\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00'
```

2.3.15 Message 2105: /peer show

The UI requests details about a peer.

Parameters

Table 2.12: Message 2105: /peer show parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12
Peer name	EOFsdt: name	Name, as known by the CS	karl-otto

Possible answers

- 1101
- 1105

Example

```
>>> import ceof
>>> cmd="2105"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> name=ceof.fillup("telmich", 128)
>>> cmd  + id + name
'2105wRHWc8telmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00'
```

2.3.16 Message 2106: /peer list

The UI requests the list of known peers.

Parameters

Table 2.13: Message 2106: /peer list parameters

Parameter	**Type**	**Description**	**Example**
ID	EOFsdt: id	packet id	afdb12

Possible answers

- 1101
- 1104

Example

```
>>> import ceof
>>> cmd="2106"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> cmd + id
'2106LGMsYS'
```

2.3.17 Message 2107: /peer send

The UI wants to submit a message to a peer.

Parameters

Table 2.14: Message 2103: /peer send parameters

Parameter	**Type**	**Description**	**Example**
ID	EOFsdt: id	packet id	afdb12
Peer name	EOFsdt: name	Name you identify the peer with	telmich
Message	EOFsdt: msgtxt	The message itself	Hallo, wie geht es Dir?

Possible answers

- 1100
- 1101

Example

```
>>> import ceof
>>> cmd="2107"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> name=ceof.fillup("telmich", 128)
>>> message=ceof.fillup("Hallo, telmich!", 256)
>>> cmd + id + name + message
'2107NassLAtelmich\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00Hallo, telmich!\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

2.3.18 Message 2199: /allquit

The user interface requests that the CS and all other UIs exit.

Parameters

Table 2.15: Message 2199: /quit parameters

Parameter	Type	Description	Example
ID	EOFsdt: id	packet id	afdb12

Possible answers

- 1101
- 1102

Example

```
>>> import ceof
>>> cmd="2199"
>>> eofid = ceof.EOFID()
>>> id = eofid.get_next()
>>> cmd + id
'2199ff5N-j'
```

Chapter 3

Implementation

The implementation is realised with *Python 3* and makes use of the *ncurses* library for drawing the user interface. The chat server (CS) and the chat ui (UI) have been integrated into the **ceof** project, which aims to provide a *secure, peer-to-peer, decentralised anonymous chat system.* For this reason, the CS and the UI are implemented as subcommands of ceof:

```
(python-env)[10:16] brief:src% ./bin/ceof
usage: ceof [-h] [-d] [-v] [-c CONFIG_DIR] [-V]
            {crypto,listener,noise,onion,peer,server,tp,ui,uiserver} ...
ceof: error: too few arguments
```

3.1 Source Code Design and Location

The implementation is made in a typical Python way: The ui and the chat server are implemented in modules, which are included into the main program.

The chat server is implemented in `src/lib/ceof/server/ui.py`, the user interface is implemented in `src/lib/ceof/ui/`.

3.2 Command line

3.2.1 Chat Server

The chat server can change the listen address and the listen port. Both may be specified on the command line:

```
(python-env)[10:16] brief:src% ./bin/ceof uiserver -h
usage: ceof uiserver [-h] [-d] [-v] [-c CONFIG_DIR] [-a ADDRESS] [-p PORT]

optional arguments:
  -h, --help            show this help message and exit
  -d, --debug           Set log level to debug
  -v, --verbose         Set log level to info, be more verbose
  -c CONFIG_DIR, --config-dir CONFIG_DIR
```

```
                        Select configuration directory ($HOME/.ceof by
                        default)
  -a ADDRESS, --address ADDRESS
                        Listen on this address for UI connections
  -p PORT, --port PORT  Listen on this port for UI connections

Get ceof at http://www.nico.schottelius.org/software/ceof/
```

The chat server can be started without any arguments:

```
(python-env)[10:33] brief:src% ./bin/ceof uiserver
```

3.2.2 User interface

The user interface accepts address and port information for the chat server to connect to on the command line:

```
(python-env)[19:39] brief:src% ./bin/ceof ui -h
usage: ceof ui [-h] [-d] [-v] [-c CONFIG_DIR] [-a ADDRESS] [-p PORT]

optional arguments:
  -h, --help            show this help message and exit
  -d, --debug           Set log level to debug
  -v, --verbose         Set log level to info, be more verbose
  -c CONFIG_DIR, --config-dir CONFIG_DIR
                        Select configuration directory ($HOME/.ceof by
                        default)
  -a ADDRESS, --address ADDRESS
                        Address to connect to
  -p PORT, --port PORT  Port to connect to

Get ceof at http://www.nico.schottelius.org/software/ceof/
```

The user interface can be started without any arguments:

```
(python-env)[10:33] brief:src% ./bin/ceof ui
```

3.3 Setup

3.3.1 Retrieve Source Code

The source code of this project can be found on `http://git.schottelius.org/?p=hszt/bachelorthesis` and can be downloaded using git:

```
git clone git://git.schottelius.org/hszt/bachelorthesis
```

3.3.2 Install Requirements

The following requirements need to be provided to run the CS and the UI:

- Python >= 3.2
- python-gnupg module (required by ceof)

After Python >= 3.2 has been installed, the following steps are necessary to get the UI and CS running:

```
# Go to home directory
cd ~

# Get source code
git clone git://git.schottelius.org/hszt/bachelorthesis

# Create python virtualenv with Python3
virtualenv3 ~/ceof-virtualenv

# Create link to python3
cd ~/ceof-virtualenv/bin
ln -s python python3

# Activate virtualenv
. ~/ceof-virtualenv/bin/activate

# Install gnupg
pip install python-gnupg
```

3.3.3 How to Run the Server and the UI

After the python environment has been setup, the server and UI can be started as following:

```
# Start Chat Server
~/bachelorarbeit/src/bin/ceof uiserver &

# Start UI
~/bachelorarbeit/src/bin/ceof ui
```

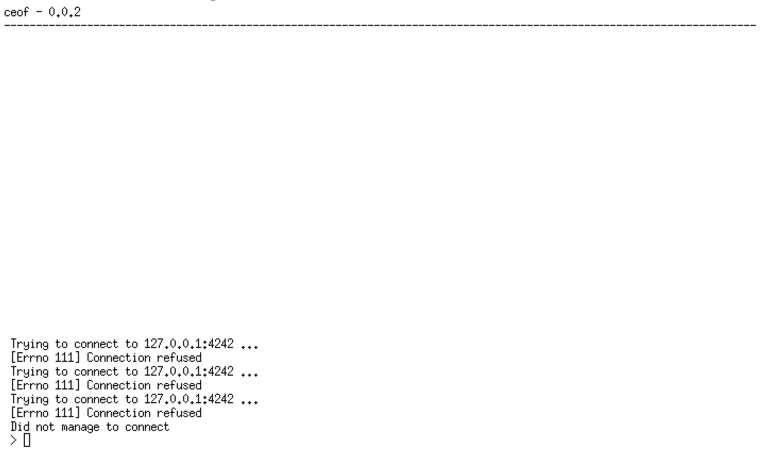
3.4 The User Interface ("'ui2user"')

This section specifies the appereance of the user interface to the user.

3.4.1 Interface

The UI is running as a ncurses application and prompts for input on a specific line.[1] All commands start with "/".

Figure 3.1: UI Startup Screen



```
ceof - 0.0.2
------------------------------------------------------------------------------------------------------------------------



 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Did not manage to connect
 > 
```

3.4.2 UI Command: /help

The /help command prints a short usage description.[2]

Example

```
/help
```

3.4.3 UI Command: /connect [host] [port]

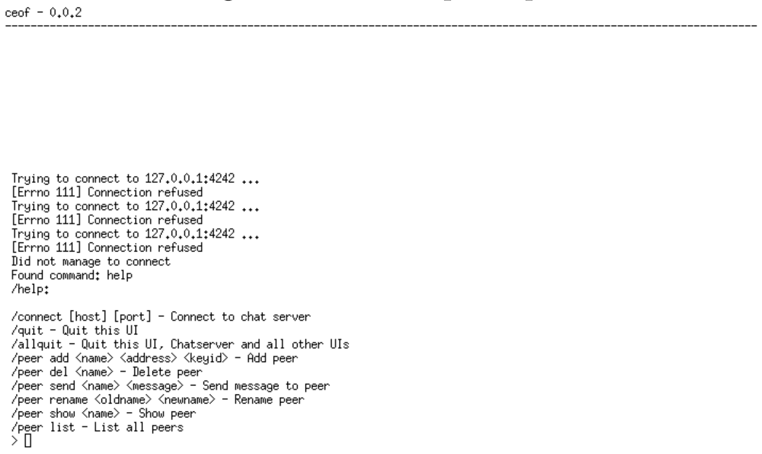
The connect command can be used to connect to the chat server.[3] Host and port are optional. If omitted, the saved host and/or port will be used. This command uses message 2100.

Example

```
/connect 127.0.0.1 4242
```

[1]Example output can be found in figure 3.1.
[2]Example output can be found in figure 3.2.
[3]Example output can be found in figure 3.3.

Figure 3.2: UI Help Output

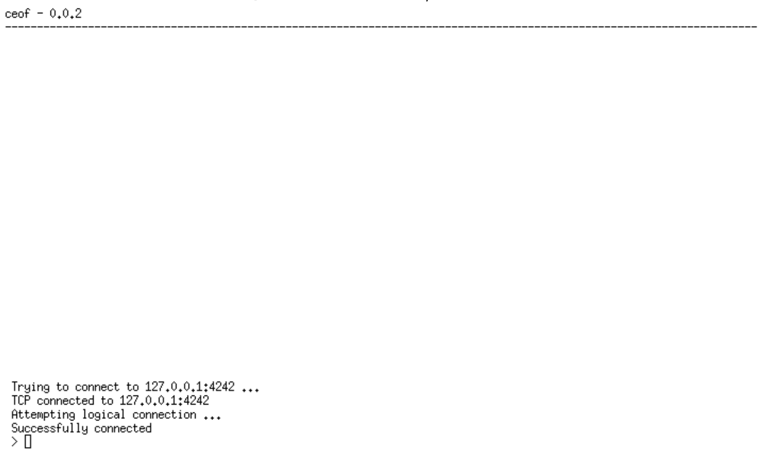


```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------------

 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Trying to connect to 127.0.0.1:4242 ...
 [Errno 111] Connection refused
 Did not manage to connect
 Found command: help
 /help:

 /connect [host] [port] - Connect to chat server
 /quit - Quit this UI
 /allquit - Quit this UI, Chatserver and all other UIs
 /peer add <name> <address> <keyid> - Add peer
 /peer del <name> - Delete peer
 /peer send <name> <message> - Send message to peer
 /peer rename <oldname> <newname> - Rename peer
 /peer show <name> - Show peer
 /peer list - List all peers
 >
```

Figure 3.3: UI /connect



```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------------

 Trying to connect to 127.0.0.1:4242 ...
 TCP connected to 127.0.0.1:4242
 Attempting logical connection ...
 Successfully connected
 >
```

3.4.4 UI Command: /quit

Request the user interface to exit. It will deregister from the CS. This command uses message 2101.

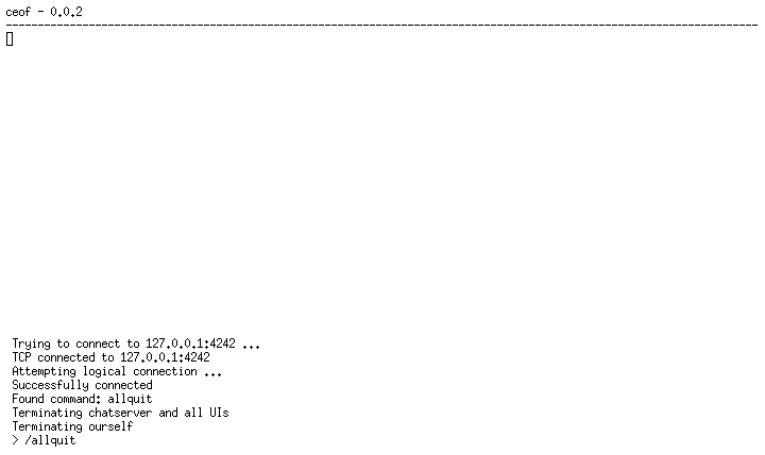
Example

```
/quit
```

3.4.5 UI Command: /allquit

The UI tells the CS and all connected UIs (including itself) to quit. This command uses message 2199.[4]

Figure 3.4: UI /allquit



```
ceof - 0.0.2
------------------------------------------------------------------------------------------------------------------
▯





 Trying to connect to 127.0.0.1:4242 ...
 TCP connected to 127.0.0.1:4242
 Attempting logical connection ...
 Successfully connected
 Found command: allquit
 Terminating chatserver and all UIs
 Terminating ourself
 > /allquit
```

Example

```
/allquit
```

3.4.6 UI Command: /peer add <name> <address> <keyid>

Add the peer with the given name *name* to the list of known peers.[5]

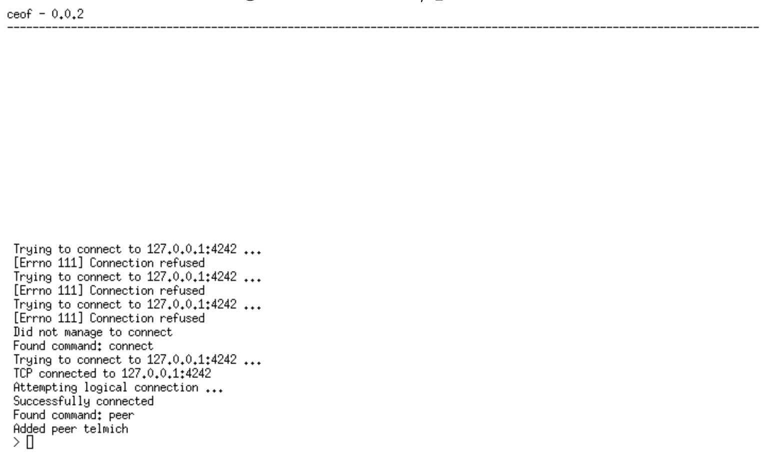
Table 3.1: UI Command: /peer add parameters

Parameter	Type	Description	Example
Peer name	EOFsdt: name	Name you identify the peer with	telmich
Address	EOFsdt: address	Where we can make the first contact	tcp://10.0.42.42:4242
Keyid	EOFsdt: keyid	The PGP fingerprint of the peers public key	F27987E34E66...

[4]Example output can be found in figure 3.4.

[5]Example output can be found in figure 3.5.

Figure 3.5: UI /peer add



```
ceof - 0.0.2
------------------------------------------------------------------------------------------------------------------

Trying to connect to 127.0.0.1:4242 ...
[Errno 111] Connection refused
Trying to connect to 127.0.0.1:4242 ...
[Errno 111] Connection refused
Trying to connect to 127.0.0.1:4242 ...
[Errno 111] Connection refused
Did not manage to connect
Found command: connect
Trying to connect to 127.0.0.1:4242 ...
TCP connected to 127.0.0.1:4242
Attempting logical connection ...
Successfully connected
Found command: peer
Added peer telmich
>
```

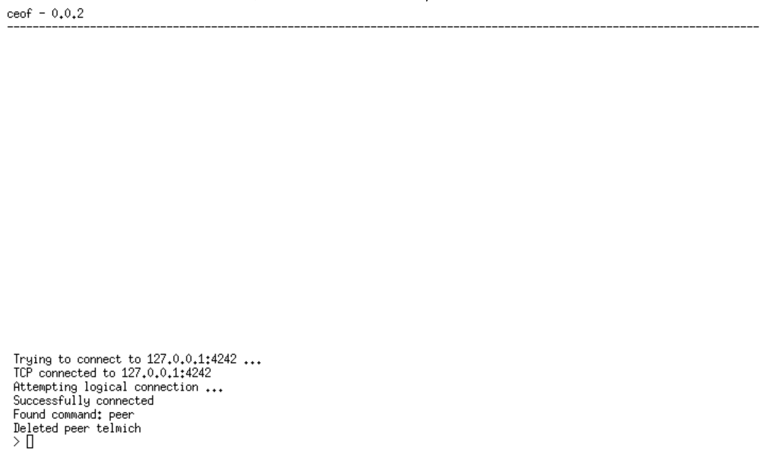
Example

```
/peer add telmich tcp//:10.0.42.42:4242 F27987E34E7866B2BA39C2FD793EB8FC325251FE
```

3.4.7 UI Command: /peer del <name>

Delete the peer with the given name *name* from the list of known peers.[6]

Figure 3.6: UI /peer del



```
ceof - 0.0.2
------------------------------------------------------------------------------------------------------------------

Trying to connect to 127.0.0.1:4242 ...
TCP connected to 127.0.0.1:4242
Attempting logical connection ...
Successfully connected
Found command: peer
Deleted peer telmich
>
```

Table 3.2: UI Command: /peer del parameters

Parameter	Type	Description	Example
Peer name	EOFsdt: name	Name you identify the peer with	telmich

Example

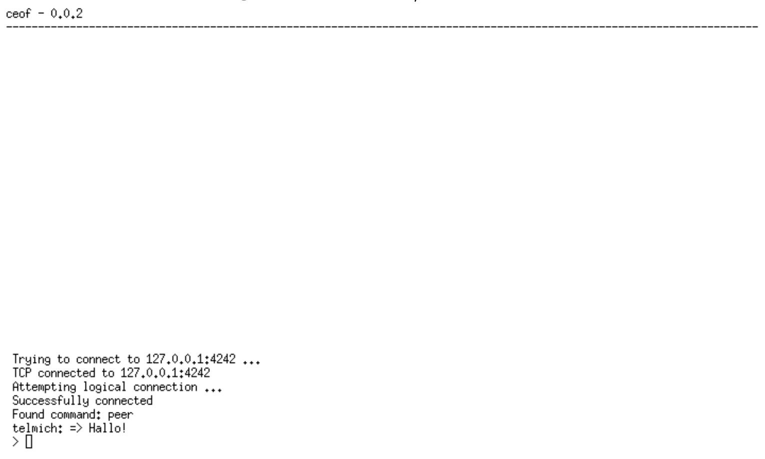
```
/peer del telmich
```

3.4.8 UI Command: /peer send <name> <msgtext>

Send message *msgtext* to peer *name*.[7]

[6]Example output can be found in figure 3.6.
[7]Example output can be found in figure 3.7.

Figure 3.7: UI /peer send



```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------

Trying to connect to 127.0.0.1:4242 ...
TCP connected to 127.0.0.1:4242
Attempting logical connection ...
Successfully connected
Found command: peer
telmich: => Hallo!
> []
```

Table 3.3: UI Command: /peer send parameters

Parameter	Type	Description	Example
Name	EOFsdt: name	Name you identify the peer with	telmich
Msgtext	EOFsdt: msgtxt	The message itself	Hallo, wie geht es Dir?

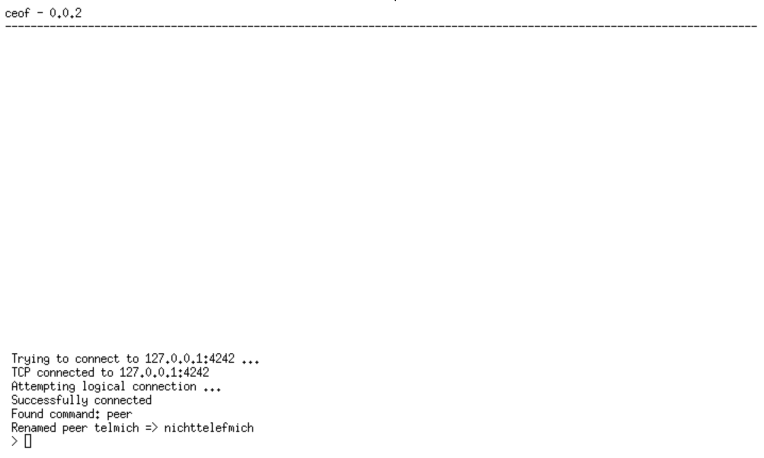
Example

```
/peer send telmich Hallo, wie geht es Dir?
```

3.4.9 UI Command: /peer rename <oldname> <newname>

Renames the peer.[8]

Figure 3.8: UI /peer rename



```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------



 Trying to connect to 127.0.0.1:4242 ...
 TCP connected to 127.0.0.1:4242
 Attempting logical connection ...
 Successfully connected
 Found command: peer
 Renamed peer telmich => nichttelefmich
 > 
```

Table 3.4: UI Command: /peer rename parameters

Parameter	Type	Description	Example
Oldname	EOFsdt: name	Old name	susi
Newname	EOFsdt: name	New name	heinz

Example

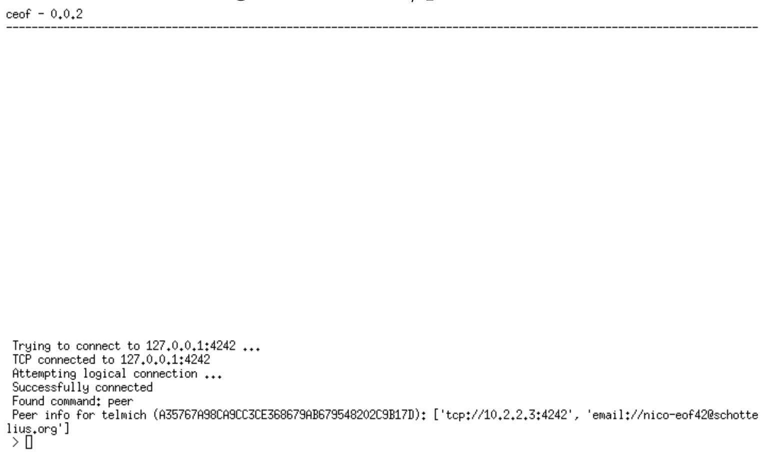
```
/peer rename susi heinz
```

3.4.10 UI Command: /peer show <name>

Display detailled information about peer.[9]

[8]Example output can be found in figure 3.8.
[9]Example output can be found in figure 3.9.

Figure 3.9: UI /peer show



```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------

 Trying to connect to 127.0.0.1:4242 ...
 TCP connected to 127.0.0.1:4242
 Attempting logical connection ...
 Successfully connected
 Found command: peer
 Peer info for telmich (A35767A98CA9CC3CE368679AB679548202C9B17D): ['tcp://10.2.2.3:4242', 'email://nico-eof42@schotte
lius.org']
 > ▯
```

Table 3.5: UI Command: /peer rename parameters

Parameter	Type	Description	Example
Peer name	EOFsdt: name	Name as known by CS	karl-otto

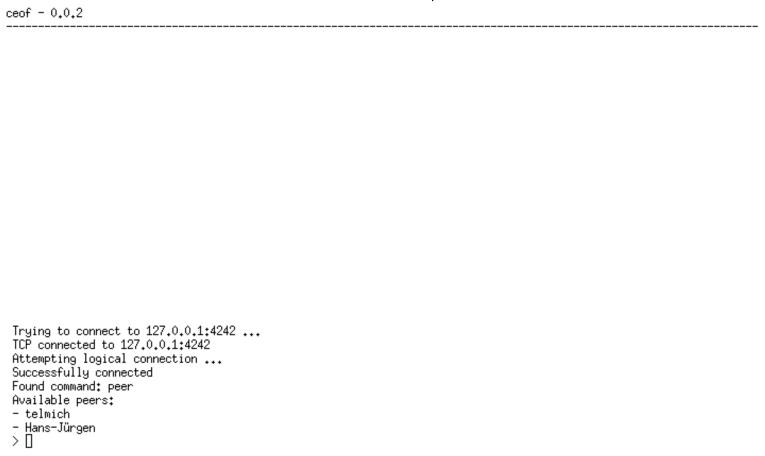
Example

```
/peer show karl-otto
```

3.4.11 UI Command: /peer list

List of currently known peers. This command does not accept any parameters.[10]

Figure 3.10: UI /peer list



```
ceof - 0.0.2
--------------------------------------------------------------------------------------------------------------------

 Trying to connect to 127.0.0.1:4242 ...
 TCP connected to 127.0.0.1:4242
 Attempting logical connection ...
 Successfully connected
 Found command: peer
 Available peers:
 - telmich
 - Hans-Jürgen
 > ▯
```

Example

```
/peer list
```

[10]Example output can be found in figure 3.10.

3.5 Conclusions

All target objectives as described in the introduction have been reached. The user interface is running as a Python/Ncurses based library, the chat protocol is based on a TCP connection with a fixed length string protocol. There are future tasks available that may enhance the usability of the UI:

- Support for command history (arrow up / down)
- Extend chat server from mockup to real implementation
- Support for line editing (delete / backspace, arrow left / right)
- Support for resynchronisation in chat server and ui

In addition to the expected results, all commands supported in the chat server have been verified by using unit testing.

Bibliography

[1] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifier (URI): generic syntax. RFC 3986, Internet Engineering Task Force, January 2005.

[2] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. RFC 2440, Internet Engineering Task Force, November 1998.