

gui2ceof

telmich

2007-10-01 v0.2-bed

Contents

1	Introduction	2
1.1	Changelog	2
1.1.1	draft-1 to v0.2-mrmcd110b	2
2	Connection	2
3	Commands	3
3.1	0000: Register client	3
3.2	0001: Deregister client	3
3.3	0009: Request for exit	4
3.4	0010: List connected markschreiers	4
3.5	0011: Connect to markschreier	4
3.6	0020: Retrieve list of known peers	4
3.7	0021: Add peer	5
3.8	0022: Send message to peer	5
3.9	0030: Get list of channels	5
3.10	0031: Ask to join a channel via marktschreier	6
3.11	0032: Send message to channel	6
3.12	0033: Create channel	6
3.13	0034: Submit channel to marktschreier	6
3.14	1300: Recieved message	7
3.15	1399: Exit notify	7
4	The way it works	7
4.1	GUI initiated commands	7
4.1.1	GUI startup	7
4.1.2	Creating a channel	8
4.1.3	Finding a channel	8

4.1.4	Joining a channel	8
4.1.5	Leaving a channel	8
4.1.6	Invite to channel	8
4.1.7	Listing friends	9
4.2	ceof initiated commands / messages	9
4.2.1	Recieved a join request	9
4.2.2	Recieved a join notification request	9
4.2.3	Recieved a channel message	9
4.2.4	Recieved a private message	9

1 Introduction

This document specifies the commands send from GUI to and from ceof¹ to the GUIs.

1.1 Changelog

1.1.1 draft-1 to v0.2-mrmcd110b

- Changed socket location
- Clarified what todo without *CEOF_DIR* and *HOME* set
- Added exit command.
- Changed number of bytes for marktschreier from 512 Bytes to 128 Bytes
- Changed number of bytes for channel name from 512 Bytes to 128 Bytes
- Added ranges (001-00x)

2 Connection

The client connects to a socket named *\$HOME/.ceof/clients/socket*. If the environment variable ”‘*CEOF_DIR*’” is set, *\$HOME/.ceof* should be replaced with that content. If the environment variable *CEOF_CLIENT_SOCKET* is set, ”‘*clients/socket*’” should be replaced by its content. If the environment variable *HOME* and *CEOF_DIR* is empty, you should fallback to the directory *.ceof* in the currenty directory.

¹the central EOF-1 application

3 Commands

All commands are send as 4-Byte ASCII digits (for instance "0012"). All answers and all numbers are ASCII-numbers. We **never** transmit binary numbers. **Client commands** always begin with **0** ("0042" for instance), **answers** or **notifications** from ceof begin with **1** ("1023" for instance). After each command follows individual data. The second byte indicates the type of message:

- **00**: client meta command (something that does not affect the user)
 - **000**: (De-)Initialisation
 - **001**: Marktschreier related
 - **002**: Peer related
 - **003**: Channel related
- **01**: messages
- **11**: sucess answers from ceof
- **12**: error answers from ceof
- **13**: messages / notifications initiated by ceof

3.1 0000: Register client

After the "0000" the client directly appends two ASCII digits containing the version of the client to ceof protocol it speaks. This specification uses version "02". Answers from ceof:

- **1100**: sucess, you are connected
- **1200**: version not supported

3.2 0001: Deregister client

This client derigesters from ceof. Ceof will keep on running, even if this was the last client. Answers from ceof:

- **1101**: sucess, you are disconnected

3.3 0009: Request for exit

Tells ceof to exit and to notify all clients to exit. It will not reply anything to you, but issue an exit notify to all clients, including the requesting one.

3.4 0010: List connected markschreiers

Answers from ceof:

- **1102:** list follows
 - four ASCII-digits containing number of peers ("num_peer")
 - after that follow *num_peer* peer ids:
 - * 4 ASCII Bytes containing the internal peer id
 - * 128 Byte containing the peer name (if shorter it is padded with 0-bytes)
 - * 40 Byte containing the fingerprint of the key²

3.5 0011: Connect to markschreier

After the "0011" follow 128 Bytes describing how to connect to the marktschreier ("tcp://62.65.138.66:42" for instance). If the URI is shorter than 128 bytes, the remain should be filled with 0 bytes (also known as '\0').

Answers from ceof:

- **1103:** success
 - After the 1103 follow four ASCII digits containing the internal marktschreier id
- **1210:** high level protocol (like "tcp://") not supported
- **1211:** connection could not be established

3.6 0020: Retrieve list of known peers

Answers from ceof:

- **1120:** list follows
 - four ASCII-digits containing number of peers ("num_peer")

²See RFC 2440, 11.2. Key IDs and Fingerprints

- after that follow *num_peer* peer ids:
 - * 4 ASCII Bytes containing the internal peer id
 - * 128 Byte containing the peer name (if shorter it is padded with 0-bytes)
 - * 40 Byte containing the fingerprint of the key³

3.7 0021: Add peer

After the "0021" follow 128 bytes describing how to connect to the peer. Answers from ceof:

- **1121:** success, registered peer
 - Follows four bytes containing the internal peer id
- **1220:** unknown peer

3.8 0022: Send message to peer

After the "0022" follow four bytes for the ceof internal peer id and after that 512 Bytes for the message. Answers from ceof:

- **1122:** success
- **1221:** unknown peer

3.9 0030: Get list of channels

After the "0030" follow 4 ASCII Bytes containing the internal markschreier id.

Answers from ceof:

- **1130:** got list; following four ASCII-digits containing number of channels ("num_chan"); after that num_chan 128 Bytes packets follow containing the channel name, padded with 0-bytes
- **1230:** connection could not be established

³See RFC 2440, 11.2. Key IDs and Fingerprints

3.10 0031: Ask to join a channel via marktschreier

After the "0031" follow

1. 4 Bytes containing the internal marktschreier id
2. 128 Bytes describing the channel name.

Answers from ceof:

- **1131:** success (means: marktschreier asked known peers to connect to us)
 - 4 Bytes follow, giving the internal channel id
- **1231:** connection could not be established
- **1232:** access denied by marktschreier: you are not allowed to join

3.11 0032: Send message to channel

After the "0032" follow four bytes for the ceof internal channel id and after that 512 Bytes for the message. Answers from ceof:

- **1132:** success
- **1233:** unknown channel

eof

3.12 0033: Create channel

After the "0033" follow 128 bytes containing the name of the channel.

- **1133:** success
 - Follows 4 Bytes containing the internal channel id
- **1234:** Channel already exists

3.13 0034: Submit channel to marktschreier

After the "0034" follow 4 bytes containing the internal channel id.

- **1134:** success
- **1235:** Channel already exists

3.14 1300: Recieved message

After the "1300" follows:

- 4 Byte channel id; if the id is "9999" it is a private message, directly send to you
- 4 Byte peer id (the person who send the message)
- 512 Byte message

3.15 1399: Exit notify

ceof is being shutdown. Shutdown yourself, too. After that message ceof will exit and you should do the same. No answer possible, ceof already decided to vanish.

4 The way it works

This section explains how the commands relate together and which commands to use in which order.

4.1 GUI initiated commands

This section and all subsections are not yet finished. They are in this draft to show the interested reader a preview of the content of the next draft.

4.1.1 GUI startup

What todo, when the GUI starts. Connect to ceof. Find out about

- joined channels,
- connected marktscheier
- and open queries.

It thus issues the following commands: register, list joined channels, list open queries, list marktschreier.

4.1.2 Creating a channel

When you want to create a channel, you simply have to give it a name. Ceof will use that name and sign it with your pgp key. The result is the global unique channel identifier. For testing, you can build your channels easily on the commandline:

```
% echo -n '!eof' > CHANNELNAME
% cat CHANNELNAME 124byteszero > CHANNELNAME.padded
% gpg -s CHANNELNAME
```

You need a passphrase to unlock the secret key for
user: "Nico Schottelius (telmich) <nico-public@schottelius.org>"
1024-bit DSA key, ID 9885188C, created 2006-09-27

```
% ls -l CHANNELNAME*
-rw----- 1 nico nico 4 2007-09-17 21:32 CHANNELNAME
-rw----- 1 nico nico 128 2007-09-17 21:32 CHANNELNAME.padded
-rw----- 1 nico nico 113 2007-09-17 21:32 CHANNELNAME.padded.gpg

% gpg -d CHANNELNAME.padded.gpg 2>/dev/null
!eof
```

4.1.3 Finding a channel

replace internal id with signature!

The GUI asks ceof, which channels are known. After that the GUI can use the internal ID (which is "unique", as in: it is a 32 bit integer that is increased as long as ceof is running) to join it.

4.1.4 Joining a channel

The channel must be known through

4.1.5 Leaving a channel

Sends a message to all known participants that we leave the channel now.

4.1.6 Invite to channel

If you create a new channel, you may want to invite people to it.

4.1.7 Listing friends

4.2 ceof initiated commands / messages

4.2.1 Recieved a join request

Somebody wants to join in a channel, in which you are a member.

4.2.2 Recieved a join notification request

Somebody wants us to tell all members of the channel that she wants to join us.

4.2.3 Recieved a channel message

Must contain which channel, who send it and what is in the message.

4.2.4 Recieved a private message

This messages was send only to us, not to a channel.