Hachat Documentation

{rohrer, dittler, wedel}@informatik.hu-berlin.de

January 27, 2013

Contents

1 Module gui

this is the module for the GUI

1.1 Variables

Name	Description
CODEC	**Value:** `'utf8'`
__package__	**Value:** `None`

1.2 Class gui

object
└ **gui.gui**

GUI Class for Hachat User Interface

1.2.1 Methods

__init__(*self*, *parent*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

check_queue(*self*)

resieve a txt-msg and push it in the gui uses a Queue which is checked every 50ms

ende(*self*)

stops hachat and gui

ende_(*self*, *event*)

event: calls ende() to stop gui and hachat

run(*self*)

start gui

senden(*self*)

send a txt-msg from the gui

senden_(*self*, *event*)

event: calls senden() to send a txt-msg from the gui

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.2.2 Properties

Name	Description
Inherited from object	
__class__	

2 Module hachat

This is the basic hachat module which starts a Peer with the given options from command line

2.1 Variables

Name	Description
__package__	**Value:** `None`
args	**Value:** `Namespace(Test=False, ip=None, link=None, name=None, port...`
name	**Value:** `'user43075'`
nr	**Value:** `43075`
parser	**Value:** `ArgumentParser(prog='(imported)', usage=None, description...`

3 Module host

This module provides the Host Class which represents connections to other Peers in a Hachat-Peer

3.1 Variables

Name	Description
__package__	**Value:** `None`

3.2 Class Host

Class representing a connection to another peer

3.2.1 Methods

__del__(*self*)

__init__(*self*, *peer*, *hostIP*, *hostPort*)

addToMsgQueue(*self*, *msg*)

check if message is type Message and add to Queue

constructKey(*cls*, *hostIP*, *hostPort*)

Class method: construct key to identify hosts in hostlist

sendHello(*self*)

will send a HELO-Message to the corresponding peer

4 Module message

module which provides all message types for hachat

4.1 Functions

toMessage(*string*)
construct Message type from string

4.2 Variables

Name	Description
__package__	**Value:** `None`

4.3 Class ByeMessage

object
message.Message
message.ByeMessage

Message Type to be send when leaving

4.3.1 Methods

__init__(*self*, *origin*, *uid*=`None`)
build Message with supplied uid or otherwise get a random uid Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)
implements interface Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),

__repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.3.2 Properties

Name	Description
Inherited from object	
__class__	

4.4 Class DeadMessage

object
└
message.Message
└
message.DeadMessage

Message that tells a Peer that another Peer is dead

4.4.1 Methods

__init__(*self*, *origin*, *peer*, *uid*=`None`)

build Message with supplied uid or otherwise get a random uid

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

implements interface

Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.4.2 Properties

Name	Description
Inherited from object	
__class__	

4.5 Class HeloMessage

regularly sent HELO Message which exchange information on IP and Port Message layout: | type | uid | recipientIP | recipientPort | senderIP | senderPort |

4.5.1 Methods

__init__(*self*, *recipientIP*, *recipientPort*, *senderIP*, *senderPort*, *uid*=`None`) build Message with supplied uid or otherwise get a random uid Overrides: object.__init__ extit(inherited documentation)

__str__(*self*) implements interface Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.5.2 Properties

Name	Description
Inherited from object	
__class__	

4.6 Class History

Klasse History speichert und ueberprueft Text-Msgs

4.6.1 Methods

__init__(*self*, *msgLimit*, *hashLimit*)

addMsg(*self*, *msg*)

add TXTMessage to History

getListMsgObjects(*self*, *msgQuant*)

get a list of x Messages

getMsgHashes(*self*, *msgQuant*)

get a list of hashes from x Messages

getMsgObjects(*self*, *msgHash*)

get Message object from hash

msgExists(*self*, *msghash*)

Message is already in History (test on hash)

msgSafed(*self*, *msg*)

Message is already in History (test on TMssage object)

removeMsg(*self*, *msgQuant*=0, *hashQuant*=0)

remove x Messages from History

4.7 Class HistoryExchangeMessage

for request and pushing Hosts Message layout: | type | uid | recipientIP | recipientPort | origin key | level | quant | liste |

4.7.1 Methods

__init__(*self*, *recipientIP*, *recipientPort*, *origin*, *level*, *quant*=`None`, *liste*=`None`, *uid*=`None`)

build Message with supplied uid or otherwise get a random uid

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

implements interface

Overrides: object.__str__

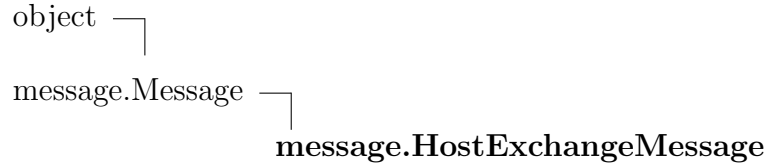
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.7.2 Properties

Name	Description
Inherited from object	
__class__	

4.8 Class HostExchangeMessage



for request and pushing Hosts Message layout: | type | uid | recipientIP | recipientPort | origin key | level | quant | listofHosts |

4.8.1 Methods

__init__(*self*, *recipientIP*, *recipientPort*, *origin*, *level*, *quant*=`None`, *listofHosts*=`None`, *uid*=`None`)

build Message with supplied uid or otherwise get a random uid

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

implements interface

Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.8.2 Properties

Name	Description
Inherited from object	
__class__	

4.9 Class Message

Known Subclasses: message.ByeMessage, message.DeadMessage, message.HeloMessage, message.HistoryExchangeMessage, message.HostExchangeMessage, message.TextMessage

abstract class all other message types will inherit from

4.9.1 Methods

__init__(*self*, *uid*)

build Message with supplied uid or otherwise get a random uid

Overrides: object.__init__

__**str**__(*self*)
cast Message to string
Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.9.2 Properties

Name	Description
Inherited from object	
__class__	

4.10 Class MessageException

object
└
exceptions.BaseException
└
exceptions.Exception
└
message.MessageException

Custom Exception Type for Messages

4.10.1 Methods

__**init**__(*self*, *value*)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

__**str**__(*self*)
str(x)
Overrides: object.__str__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

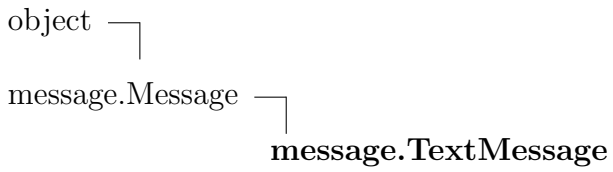
Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

4.10.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

4.11 Class TextMessage



normal Text Messages Message layout: | type | uid | hash | sender name | origin key | lastHop key | text |

4.11.1 Methods

__init__(*self*, *name*, *origin*, *lastHop*, *text*, *uid*=`None`)

build Message with supplied uid or otherwise get a random uid

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

implements interface

Overrides: object.__str__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

4.11.2 Properties

Name	Description
Inherited from object	
__class__	

5 Module peer

central module which defines the behaviour of a Hachat Peer

5.1 Variables

Name	Description
__package__	**Value:** `None`

5.2 Class Peer

Peer Klasse

5.2.1 Methods

HistoryControl(*self*, *neighbour*, *historyList*)

checks if the historyList from neighbour contains msgs wich are not in own History. If so, these lostMsg Hahses will be pushed back and by this the associated msgObjects are requested.

__del__(*self*)

__init__(*self*, *firstHost*=`None`, *port*=`None`, *name*=`'temp'`, *ip*=`None`, *testmode*=`False`)

addToHosts(*self*, *addr*)

check if already in hostlist otherwise add

forwardMsg(*self*, *msg*, *Oneneigbour*=`None`)

forwarding TextMessage, but not to initial sender if host is set, it will only forward to this single host

generateMsgParts(*self*, *quant*=`5`, *length*=`2000`)

generates random TextMsgs, if length > 1000 there will be more then one msg-part

getHistory(*self*, *neighbour*, *initial*=`False`)

request History from neigbour; initial is true for a initial history exchange: this will skip pushing Hashes and immediately request msg.objects

maintenanceLoop(*self*)

thread which runs regularly maintenance tasks

processMessage(*self*, *msg*, *fromAddr*)

processes the received messages

pushHistory(*self*, *neighbour*, *quant*)

push own List of History-Hashes to neighbour

pushHosts(*self*, *neighbour*, *quant*)

give Hosts from hostExchange to a neighbour

pushMsgObjects(*self*, *neighbour*, *lostMsgHashes*=`None`)

pushes requested msgObjects back to neighbour

requestHosts(*self*, *neighbour*, *quant*=`None`)

request Hosts from neighbour

sendAll(*self*, *msg*)

send Message Object to all your Peers

sendLoop(*self*, *test*=`False`)

send Message objects of all hosts from Queue as string

sendText(*self*, *text*)

make TXTMessage out of text and send to all hosts

startRecvLoop(*self*)

general receive loop of a peer

Index