

PythonDHCP

Generated by Doxygen 1.9.3

1 Python DHCP Server	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 dhcp Namespace Reference	11
6.1.1 Function Documentation	12
6.1.1.1 debug_msg()	12
6.1.1.2 get_host_ip_addresses()	12
6.1.1.3 ip_addresses()	12
6.1.1.4 sorted_hosts()	12
6.1.2 Variable Documentation	12
6.1.2.1 ALL	13
6.1.2.2 configuration	13
6.1.2.3 debug	13
6.1.2.4 ip_address_lease_time	13
6.1.2.5 messages	13
6.1.2.6 server	13
6.2 listener Namespace Reference	14
6.2.1 Function Documentation	14
6.2.1.1 inet_atonX()	14
6.2.1.2 inet_ntoaX()	14
6.2.1.3 macpack()	15
6.2.1.4 macunpack()	15
6.2.1.5 packbool()	15
6.2.1.6 unpackbool()	15
6.2.2 Variable Documentation	15
6.2.2.1 data	15
6.2.2.2 dhcp_message_types	16
6.2.2.3 options	16
6.2.2.4 p	16
6.2.2.5 packet	16
6.2.2.6 reads	16
6.2.2.7 reversed_dhcp_message_types	17

6.2.2.8 s1	17
6.2.2.9 shortpack	17
6.2.2.10 shortunpack	17
6.3 tldict Namespace Reference	17
7 Class Documentation	19
7.1 dhcp.ALL Class Reference	19
7.1.1 Detailed Description	19
7.1.2 Member Function Documentation	19
7.1.2.1 __eq__()	19
7.1.2.2 __repr__()	20
7.2 dhcp.CASEINSENSITIVE Class Reference	20
7.2.1 Detailed Description	20
7.2.2 Constructor & Destructor Documentation	20
7.2.2.1 __init__()	20
7.2.3 Member Function Documentation	21
7.2.3.1 __eq__()	21
7.2.4 Member Data Documentation	21
7.2.4.1 s	21
7.3 dhcp.CSVDatabase Class Reference	21
7.3.1 Detailed Description	22
7.3.2 Constructor & Destructor Documentation	22
7.3.2.1 __init__()	22
7.3.3 Member Function Documentation	22
7.3.3.1 add()	22
7.3.3.2 all()	23
7.3.3.3 delete()	23
7.3.3.4 file()	23
7.3.3.5 get()	23
7.3.4 Member Data Documentation	24
7.3.4.1 delimiter	24
7.3.4.2 file_name	24
7.4 dhcp.DHCP Server Class Reference	24
7.4.1 Detailed Description	25
7.4.2 Constructor & Destructor Documentation	25
7.4.2.1 __init__()	25
7.4.3 Member Function Documentation	25
7.4.3.1 broadcast()	25
7.4.3.2 client_has_chosen()	25
7.4.3.3 close()	26
7.4.3.4 debug_clients()	26
7.4.3.5 get_all_hosts()	26

7.4.3.6 <code>get_current_hosts()</code>	26
7.4.3.7 <code>get_ip_address()</code>	26
7.4.3.8 <code>is_valid_client_address()</code>	27
7.4.3.9 <code>received()</code>	27
7.4.3.10 <code>run()</code>	27
7.4.3.11 <code>run_in_thread()</code>	27
7.4.3.12 <code>server_identifiers()</code>	27
7.4.3.13 <code>update()</code>	28
7.4.4 Member Data Documentation	28
7.4.4.1 <code>closed</code>	28
7.4.4.2 <code>configuration</code>	28
7.4.4.3 <code>delay_worker</code>	28
7.4.4.4 <code>hosts</code>	28
7.4.4.5 <code>socket</code>	29
7.4.4.6 <code>time_started</code>	29
7.4.4.7 <code>transactions</code>	29
7.5 <code>dhcp.DHCPServerConfiguration</code> Class Reference	29
7.5.1 Detailed Description	30
7.5.2 Member Function Documentation	30
7.5.2.1 <code>adjust_if_this_computer_is_a_router()</code>	30
7.5.2.2 <code>all_ip_addresses()</code>	30
7.5.2.3 <code>load()</code>	31
7.5.2.4 <code>network_filter()</code>	31
7.5.3 Member Data Documentation	31
7.5.3.1 <code>broadcast_address</code> [1/2]	31
7.5.3.2 <code>broadcast_address</code> [2/2]	31
7.5.3.3 <code>debug</code>	31
7.5.3.4 <code>dhcp_acknowledge_after_seconds</code>	32
7.5.3.5 <code>dhcp_offer_after_seconds</code>	32
7.5.3.6 <code>domain_name_server</code>	32
7.5.3.7 <code>host_file</code>	32
7.5.3.8 <code>ip_address_lease_time</code>	32
7.5.3.9 <code>length_of_transaction</code>	32
7.5.3.10 <code>network</code> [1/2]	33
7.5.3.11 <code>network</code> [2/2]	33
7.5.3.12 <code>router</code>	33
7.5.3.13 <code>subnet_mask</code>	33
7.6 <code>dhcp.DHCPTransaction</code> Class Reference	33
7.6.1 Detailed Description	34
7.6.2 Constructor & Destructor Documentation	34
7.6.2.1 <code>__init__()</code>	34
7.6.3 Member Function Documentation	34

7.6.3.1 acknowledge()	35
7.6.3.2 close()	35
7.6.3.3 is_done()	35
7.6.3.4 receive()	35
7.6.3.5 received_dhcp_discover()	36
7.6.3.6 received_dhcp_inform()	36
7.6.3.7 received_dhcp_request()	36
7.6.3.8 send_offer()	36
7.6.4 Member Data Documentation	37
7.6.4.1 configuration	37
7.6.4.2 debug	37
7.6.4.3 do_after	37
7.6.4.4 done	37
7.6.4.5 done_time	37
7.6.4.6 packets	38
7.6.4.7 server	38
7.7 dhcp.GREATER Class Reference	38
7.7.1 Detailed Description	38
7.7.2 Constructor & Destructor Documentation	38
7.7.2.1 __init__()	39
7.7.3 Member Function Documentation	39
7.7.3.1 __eq__()	39
7.7.4 Member Data Documentation	39
7.7.4.1 value	39
7.8 dhcp.Host Class Reference	39
7.8.1 Detailed Description	40
7.8.2 Constructor & Destructor Documentation	40
7.8.2.1 __init__()	40
7.8.3 Member Function Documentation	40
7.8.3.1 __eq__()	41
7.8.3.2 __hash__()	41
7.8.3.3 from_packet()	41
7.8.3.4 from_tuple()	41
7.8.3.5 get_pattern()	41
7.8.3.6 has_valid_ip()	42
7.8.3.7 to_pattern()	42
7.8.3.8 to_tuple()	42
7.8.4 Member Data Documentation	42
7.8.4.1 hostname	42
7.8.4.2 ip	43
7.8.4.3 last_used	43
7.8.4.4 mac	43

7.9 dhcp.HostDatabase Class Reference	43
7.9.1 Detailed Description	44
7.9.2 Constructor & Destructor Documentation	44
7.9.2.1 __init__()	44
7.9.3 Member Function Documentation	44
7.9.3.1 add()	44
7.9.3.2 all()	44
7.9.3.3 delete()	44
7.9.3.4 get()	45
7.9.3.5 replace()	45
7.9.4 Member Data Documentation	45
7.9.4.1 db	45
7.10 dhcp.NETWORK Class Reference	45
7.10.1 Detailed Description	46
7.10.2 Constructor & Destructor Documentation	46
7.10.2.1 __init__()	46
7.10.3 Member Function Documentation	46
7.10.3.1 __eq__()	46
7.10.4 Member Data Documentation	46
7.10.4.1 network	46
7.10.4.2 subnet_mask	47
7.11 dhcp.PriorityQueue Class Reference	47
7.11.1 Detailed Description	47
7.11.2 Constructor & Destructor Documentation	47
7.11.2.1 __init__()	47
7.11.3 Member Function Documentation	48
7.11.3.1 get()	48
7.11.3.2 put()	48
7.11.3.3 qsize()	48
7.12 listener.ReadBootProtocolPacket Class Reference	48
7.12.1 Detailed Description	49
7.12.2 Constructor & Destructor Documentation	49
7.12.2.1 __init__()	50
7.12.3 Member Function Documentation	50
7.12.3.1 __contains__()	50
7.12.3.2 __getitem__()	50
7.12.3.3 __gt__()	50
7.12.3.4 __str__()	50
7.12.3.5 formatted_named_options()	51
7.12.4 Member Data Documentation	51
7.12.4.1 address	51
7.12.4.2 bootp_flags	51

7.12.4.3 CHADDR	51
7.12.4.4 CIADDR	51
7.12.4.5 client_ip_address	51
7.12.4.6 client_mac_address	52
7.12.4.7 data	52
7.12.4.8 FLAGS	52
7.12.4.9 GIADDR	52
7.12.4.10 hardware_address_length	52
7.12.4.11 hardware_type	52
7.12.4.12 HLEN	53
7.12.4.13 hops	53
7.12.4.14 HOPS	53
7.12.4.15 host	53
7.12.4.16 HTYPE	53
7.12.4.17 magic_cookie	53
7.12.4.18 message_type	54
7.12.4.19 named_options	54
7.12.4.20 next_server_ip_address	54
7.12.4.21 OP	54
7.12.4.22 options	54
7.12.4.23 port	54
7.12.4.24 relay_agent_ip_address	55
7.12.4.25 seconds_elapsed	55
7.12.4.26 SECS	55
7.12.4.27 SIADDR	55
7.12.4.28 transaction_id	55
7.12.4.29 XID	55
7.12.4.30 YIADDR	56
7.12.4.31 your_ip_address	56
7.13 dhcp.ThreadedTcpRequestHandler Class Reference	56
7.13.1 Detailed Description	56
7.13.2 Member Function Documentation	56
7.13.2.1 handle()	57
7.14 dhcp.ThreadedTcpServer Class Reference	57
7.14.1 Detailed Description	57
7.14.2 Member Function Documentation	57
7.14.2.1 setConfiguration()	58
7.14.2.2 setEvents()	58
7.14.2.3 setHosts()	58
7.14.3 Member Data Documentation	58
7.14.3.1 configuration	58
7.14.3.2 events	59

7.14.3.3 hosts	59
7.15 dhcp.TransactionDelayWorker Class Reference	59
7.15.1 Detailed Description	59
7.15.2 Constructor & Destructor Documentation	60
7.15.2.1 __init__()	60
7.15.3 Member Function Documentation	60
7.15.3.1 close()	60
7.15.3.2 do_after()	60
7.15.4 Member Data Documentation	60
7.15.4.1 closed	61
7.15.4.2 queue	61
7.15.4.3 thread	61
7.16 ttlDict.TTLOrderedDict Class Reference	61
7.16.1 Detailed Description	62
7.16.2 Constructor & Destructor Documentation	62
7.16.2.1 __init__()	62
7.16.3 Member Function Documentation	62
7.16.3.1 __delitem__()	62
7.16.3.2 __getitem__()	62
7.16.3.3 __iter__()	63
7.16.3.4 __len__()	63
7.16.3.5 __repr__()	63
7.16.3.6 __setitem__()	63
7.16.3.7 expire_at()	63
7.16.3.8 get()	64
7.16.3.9 get_ttl()	64
7.16.3.10 is_expired()	64
7.16.3.11 items()	64
7.16.3.12 keys()	65
7.16.3.13 set_ttl()	65
7.16.3.14 values()	65
7.17 dhcp.WriteBootProtocolPacket Class Reference	65
7.17.1 Detailed Description	66
7.17.2 Constructor & Destructor Documentation	66
7.17.2.1 __init__()	66
7.17.3 Member Function Documentation	67
7.17.3.1 __str__()	67
7.17.3.2 get_option()	67
7.17.3.3 options()	67
7.17.3.4 to_bytes()	67
7.17.4 Member Data Documentation	68
7.17.4.1 bootp_flags	68

7.17.4.2 client_ip_address	68
7.17.4.3 client_mac_address	68
7.17.4.4 hardware_address_length	68
7.17.4.5 hardware_type	68
7.17.4.6 hops	69
7.17.4.7 magic_cookie	69
7.17.4.8 message_type	69
7.17.4.9 next_server_ip_address	69
7.17.4.10 parameter_order	69
7.17.4.11 relay_agent_ip_address	69
7.17.4.12 seconds_elapsed	70
7.17.4.13 transaction_id	70
7.17.4.14 your_ip_address	70
8 File Documentation	71
8.1 dhcp.py File Reference	71
8.2 dhcp.py	72
8.3 listener.py File Reference	80
8.4 listener.py	81
8.5 README.md File Reference	84
8.6 ttldict.py File Reference	84
8.7 ttldict.py	85
Index	87

Chapter 1

Python DHCP Server

This is a purely Python DHCP server that does not require any additional libraries or installs other than Python 3.

This DHCP server program will assign IP addresses ten seconds after it received packets from clients. So it can be used in networks that already have a dhcp server running.

First argument of program is tested for being configuration file eg. ./dhcp.py dhcp.conf if that file does not exist arguments are read from command line, strings must be so called double escaped "string" `dhcp.py -broadcast_address "['255.255.255.255']" -name_server "192.168.0.1"`

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

dhcp	11
listener	14
ttldict	17

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

object	
dhcp.ALL	19
dhcp.CASEINSENSITIVE	20
dhcp.CSVDatabase	21
dhcp.DHCPServer	24
dhcp.DHCPServerConfiguration	29
dhcp.DHCPTransaction	33
dhcp.GREATER	38
dhcp.Host	39
dhcp.HostDatabase	43
dhcp.NETWORK	45
dhcp.PriorityQueue	47
dhcp.TransactionDelayWorker	59
dhcp.WriteBootProtocolPacket	65
listener.ReadBootProtocolPacket	48
socketserver.StreamRequestHandler	
dhcp.ThreadedTcpRequestHandler	56
socketserver.TCPServer	
dhcp.ThreadedTcpServer	57
socketserver.ThreadingMixIn	
dhcp.ThreadedTcpServer	57
OrderedDict	
ttlDict.TTLOrderedDict	61

Chapter 4

Class Index

4.1 Class List

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

dhcp.ALL	19
dhcp.CASEINSENSITIVE	20
dhcp.CSVDatabase	21
dhcp.DHCPServer	24
dhcp.DHCPServerConfiguration	29
dhcp.DHCPTransaction	33
dhcp.GREATER	38
dhcp.Host	39
dhcp.HostDatabase	43
dhcp.NETWORK	45
dhcp.PriorityQueue	47
listener.ReadBootProtocolPacket	48
dhcp.ThreadedTcpRequestHandler	56
dhcp.ThreadedTcpServer	57
dhcp.TransactionDelayWorker	59
ttlDict.TTLOrderedDict	61
dhcp.WriteBootProtocolPacket	65

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

dhcp.py	71
listener.py	80
ttldict.py	84

Chapter 6

Namespace Documentation

6.1 dhcp Namespace Reference

Classes

- class [ALL](#)
- class [CASEINSENSITIVE](#)
- class [CSVDatabase](#)
- class [DHCPServer](#)
- class [DHCPServerConfiguration](#)
- class [DHCPTransaction](#)
- class [GREATER](#)
- class [Host](#)
- class [HostDatabase](#)
- class [NETWORK](#)
- class [PriorityQueue](#)
- class [ThreadedTcpRequestHandler](#)
- class [ThreadedTcpServer](#)
- class [TransactionDelayWorker](#)
- class [WriteBootProtocolPacket](#)

Functions

- def [get_host_ip_addresses](#) ()
- def [ip_addresses](#) (network, subnet_mask)
- def [sorted_hosts](#) (hosts)
- def [debug_msg](#) (msg, type)

Variables

- [ALL](#) = [ALL](#)()
- [messages](#) = [TTLOrderedDict](#)(default_ttl=86400)
- [configuration](#) = [DHCPServerConfiguration](#)()
- [debug](#)
- [ip_address_lease_time](#)
- [server](#) = [DHCPServer](#)(configuration)

6.1.1 Function Documentation

6.1.1.1 `debug_msg()`

```
def dhcp.debug_msg (
    msg,
    type )
```

Definition at line [716](#) of file [dhcp.py](#).

6.1.1.2 `get_host_ip_addresses()`

```
def dhcp.get_host_ip_addresses ( )
```

Get IP address of current host.

Definition at line [59](#) of file [dhcp.py](#).

6.1.1.3 `ip_addresses()`

```
def dhcp.ip_addresses (
    network,
    subnet_mask )
```

Definition at line [352](#) of file [dhcp.py](#).

6.1.1.4 `sorted_hosts()`

```
def dhcp.sorted_hosts (
    hosts )
```

Definition at line [515](#) of file [dhcp.py](#).

6.1.2 Variable Documentation

6.1.2.1 ALL

```
dhcp.ALL = ALL()
```

Definition at line 369 of file [dhcp.py](#).

6.1.2.2 configuration

```
dhcp.configuration = DHCPConfiguration()
```

Definition at line 721 of file [dhcp.py](#).

6.1.2.3 debug

```
dhcp.debug
```

Definition at line 722 of file [dhcp.py](#).

6.1.2.4 ip_address_lease_time

```
dhcp.ip_address_lease_time
```

Definition at line 726 of file [dhcp.py](#).

6.1.2.5 messages

```
dhcp.messages = TTLOrderedDict(default_ttl=86400)
```

Definition at line 714 of file [dhcp.py](#).

6.1.2.6 server

```
dhcp.server = DHCPServer(configuration)
```

Definition at line 727 of file [dhcp.py](#).

6.2.2.2 dhcp_message_types

dictionary listener.dhcp_message_types

Initial value:

```
00001 = {
00002     1 : 'DHCPDISCOVER',
00003     2 : 'DHCPOFFER',
00004     3 : 'DHCPREQUEST',
00005     4 : 'DHCPDECLINE',
00006     5 : 'DHCPACK',
00007     6 : 'DHCPNAK',
00008     7 : 'DHCPRELEASE',
00009     8 : 'DHCPINFORM',
00010 }
```

Definition at line 17 of file [listener.py](#).

6.2.2.3 options

list listener.options

Definition at line 48 of file [listener.py](#).

6.2.2.4 p

listener.p = [ReadBootProtocolPacket](#)(data)

Definition at line 226 of file [listener.py](#).

6.2.2.5 packet

listener.packet = [ReadBootProtocolPacket](#)(*s.recvfrom(4096))

Definition at line 258 of file [listener.py](#).

6.2.2.6 reads

listener.reads = select.select([s1], [], [], 1)[0]

Definition at line 256 of file [listener.py](#).

6.2.2.7 reversed_dhcp_message_types

```
listener.reversed_dhcp_message_types = dict()
```

Definition at line 27 of file [listener.py](#).

6.2.2.8 s1

```
listener.s1 = socket(type = SOCK_DGRAM)
```

Definition at line 249 of file [listener.py](#).

6.2.2.9 shortpack

```
listener.shortpack = lambda i: bytes([i >> 8, i & 255])
```

Definition at line 32 of file [listener.py](#).

6.2.2.10 shortunpack

```
listener.shortunpack = lambda data: (data[0] << 8) + data[1]
```

Definition at line 31 of file [listener.py](#).

6.3 ttldict Namespace Reference

Classes

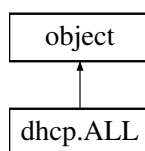
- class [TTLOrderedDict](#)

Chapter 7

Class Documentation

7.1 dhcp.ALL Class Reference

Inheritance diagram for dhcp.ALL:



Public Member Functions

- def `__eq__`(self, other)
- def `__repr__`(self)

7.1.1 Detailed Description

Comparator class

Definition at line 361 of file [dhcp.py](#).

7.1.2 Member Function Documentation

7.1.2.1 `__eq__()`

```
def dhcp.ALL.__eq__(  
    self,  
    other )
```

Definition at line 364 of file [dhcp.py](#).

7.1.2.2 `__repr__()`

```
def dhcp.ALL.__repr__ (
    self )
```

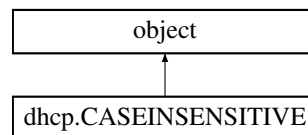
Definition at line 366 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.2 `dhcp.CASEINSENSITIVE` Class Reference

Inheritance diagram for `dhcp.CASEINSENSITIVE`:



Public Member Functions

- `def __init__ (self, s)`
- `def __eq__ (self, other)`

Public Attributes

- [s](#)

7.2.1 Detailed Description

Comparator class

Definition at line 391 of file [dhcp.py](#).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `__init__()`

```
def dhcp.CASEINSENSITIVE.__init__ (
    self,
    s )
```

Definition at line 394 of file [dhcp.py](#).

7.2.3 Member Function Documentation

7.2.3.1 `__eq__()`

```
def dhcp.CASEINSENSITIVE.__eq__ (
    self,
    other )
```

Definition at line 396 of file [dhcp.py](#).

7.2.4 Member Data Documentation

7.2.4.1 `s`

```
dhcp.CASEINSENSITIVE.s
```

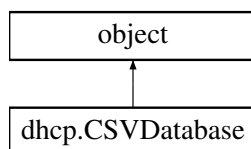
Definition at line 395 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.3 dhcp.CSVDatabase Class Reference

Inheritance diagram for dhcp.CSVDatabase:



Public Member Functions

- def `__init__` (self, [file_name](#))
- def [file](#) (self, mode='r')
- def [get](#) (self, pattern)
- def [add](#) (self, line)
- def [delete](#) (self, pattern)
- def [all](#) (self)

Public Attributes

- [file_name](#)

Static Public Attributes

- string [delimiter](#) = ';'

7.3.1 Detailed Description

Class handling CSV file database to keep host definitions

Definition at line [399](#) of file [dhcp.py](#).

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__()

```
def dhcp.CSVDatabase.__init__ (
    self,
    file_name )
```

Construct new CSV database with storage in file_name

Definition at line [404](#) of file [dhcp.py](#).

7.3.3 Member Function Documentation

7.3.3.1 add()

```
def dhcp.CSVDatabase.add (
    self,
    line )
```

Add host entry to CSV file

Definition at line [421](#) of file [dhcp.py](#).

7.3.3.2 all()

```
def dhcp.CSVDatabase.all (
    self )
```

Get all entries from CSV file

Definition at line 437 of file [dhcp.py](#).

7.3.3.3 delete()

```
def dhcp.CSVDatabase.delete (
    self,
    pattern )
```

Delete host entry from CSV file

Definition at line 427 of file [dhcp.py](#).

7.3.3.4 file()

```
def dhcp.CSVDatabase.file (
    self,
    mode = 'r' )
```

Open CSV file with selected mode

Definition at line 410 of file [dhcp.py](#).

7.3.3.5 get()

```
def dhcp.CSVDatabase.get (
    self,
    pattern )
```

Get CSV entry representing host (MAC) and lease (IP)

Definition at line 415 of file [dhcp.py](#).

7.3.4 Member Data Documentation

7.3.4.1 delimiter

```
string dhcp.CSVDatabase.delimiter = ';' [static]
```

Definition at line 402 of file [dhcp.py](#).

7.3.4.2 file_name

```
dhcp.CSVDatabase.file_name
```

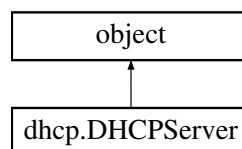
Definition at line 407 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.4 dhcp.DHCP Server Class Reference

Inheritance diagram for dhcp.DHCP Server:



Public Member Functions

- def [__init__](#) (self, [configuration](#)=None)
- def [close](#) (self)
- def [update](#) (self, timeout=0)
- def [received](#) (self, packet)
- def [client_has_chosen](#) (self, packet)
- def [is_valid_client_address](#) (self, address)
- def [get_ip_address](#) (self, packet)
- def [server_identifiers](#) (self)
- def [broadcast](#) (self, packet)
- def [run](#) (self)
- def [run_in_thread](#) (self)
- def [debug_clients](#) (self)
- def [get_all_hosts](#) (self)
- def [get_current_hosts](#) (self)

Public Attributes

- [configuration](#)
- [socket](#)
- [delay_worker](#)
- [closed](#)
- [transactions](#)
- [hosts](#)
- [time_started](#)

7.4.1 Detailed Description

Definition at line 520 of file [dhcp.py](#).

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `__init__()`

```
def dhcp.DHCP Server.__init__ (
    self,
    configuration = None )
```

Definition at line 522 of file [dhcp.py](#).

7.4.3 Member Function Documentation

7.4.3.1 `broadcast()`

```
def dhcp.DHCP Server.broadcast (
    self,
    packet )
```

Definition at line 619 of file [dhcp.py](#).

7.4.3.2 `client_has_chosen()`

```
def dhcp.DHCP Server.client_has_chosen (
    self,
    packet )
```

Definition at line 566 of file [dhcp.py](#).

7.4.3.3 close()

```
def dhcp.DHCPServer.close (
    self )
```

Definition at line 536 of file [dhcp.py](#).

7.4.3.4 debug_clients()

```
def dhcp.DHCPServer.debug_clients (
    self )
```

Definition at line 648 of file [dhcp.py](#).

7.4.3.5 get_all_hosts()

```
def dhcp.DHCPServer.get_all_hosts (
    self )
```

Definition at line 654 of file [dhcp.py](#).

7.4.3.6 get_current_hosts()

```
def dhcp.DHCPServer.get_current_hosts (
    self )
```

Definition at line 657 of file [dhcp.py](#).

7.4.3.7 get_ip_address()

```
def dhcp.DHCPServer.get_ip_address (
    self,
    packet )
```

Definition at line 581 of file [dhcp.py](#).

7.4.3.8 is_valid_client_address()

```
def dhcp.DHCP Server.is_valid_client_address (
    self,
    address )
```

Definition at line 573 of file [dhcp.py](#).

7.4.3.9 received()

```
def dhcp.DHCP Server.received (
    self,
    packet )
```

Definition at line 562 of file [dhcp.py](#).

7.4.3.10 run()

```
def dhcp.DHCP Server.run (
    self )
```

Definition at line 634 of file [dhcp.py](#).

7.4.3.11 run_in_thread()

```
def dhcp.DHCP Server.run_in_thread (
    self )
```

Definition at line 643 of file [dhcp.py](#).

7.4.3.12 server_identifiers()

```
def dhcp.DHCP Server.server_identifiers (
    self )
```

Definition at line 616 of file [dhcp.py](#).

7.4.3.13 update()

```
def dhcp.DHCPServer.update (
    self,
    timeout = 0 )
```

Definition at line [543](#) of file [dhcp.py](#).

7.4.4 Member Data Documentation

7.4.4.1 closed

```
dhcp.DHCPServer.closed
```

Definition at line [531](#) of file [dhcp.py](#).

7.4.4.2 configuration

```
dhcp.DHCPServer.configuration
```

Definition at line [526](#) of file [dhcp.py](#).

7.4.4.3 delay_worker

```
dhcp.DHCPServer.delay_worker
```

Definition at line [530](#) of file [dhcp.py](#).

7.4.4.4 hosts

```
dhcp.DHCPServer.hosts
```

Definition at line [533](#) of file [dhcp.py](#).

7.4.4.5 socket

`dhcp.DHCPServer.socket`

Definition at line 527 of file [dhcp.py](#).

7.4.4.6 time_started

`dhcp.DHCPServer.time_started`

Definition at line 534 of file [dhcp.py](#).

7.4.4.7 transactions

`dhcp.DHCPServer.transactions`

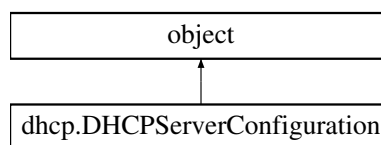
Definition at line 532 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.5 dhcp.DHCPServerConfiguration Class Reference

Inheritance diagram for `dhcp.DHCPServerConfiguration`:



Public Member Functions

- `def load (self, file)`
- `def adjust_if_this_computer_is_a_router (self)`
- `def all_ip_addresses (self)`
- `def network_filter (self)`

Public Attributes

- `network`
- `broadcast_address`

Static Public Attributes

- int `dhcp_offer_after_seconds` = 10
- int `dhcp_acknowledge_after_seconds` = 10
- int `length_of_transaction` = 40
- string `network` = '192.168.173.0'
- string `broadcast_address` = '255.255.255.255'
- string `subnet_mask` = '255.255.255.0'
- `router` = None
- int `ip_address_lease_time` = 300
- `domain_name_server` = None
- string `host_file` = 'hosts.csv'
- `debug` = lambda *args, **kw: None

7.5.1 Detailed Description

Class to load DHCP server configuration from file or command line

Definition at line 295 of file [dhcp.py](#).

7.5.2 Member Function Documentation

7.5.2.1 `adjust_if_this_computer_is_a_router()`

```
def dhcp.DHCPServerConfiguration.adjust_if_this_computer_is_a_router (
    self )
```

Automatically adjust some DHCP configuration parameters if this computer is router

Definition at line 329 of file [dhcp.py](#).

7.5.2.2 `all_ip_addresses()`

```
def dhcp.DHCPServerConfiguration.all_ip_addresses (
    self )
```

Definition at line 343 of file [dhcp.py](#).

7.5.2.3 load()

```
def dhcp.DHCPServerConfiguration.load (
    self,
    file )
```

Load configuration from file using exec to parse file as object dictionary or get ALL command line arguments and change them using regexp to file layout and treat as file

Definition at line 314 of file [dhcp.py](#).

7.5.2.4 network_filter()

```
def dhcp.DHCPServerConfiguration.network_filter (
    self )
```

Definition at line 349 of file [dhcp.py](#).

7.5.3 Member Data Documentation

7.5.3.1 broadcast_address [1/2]

```
string dhcp.DHCPServerConfiguration.broadcast_address = '255.255.255.255' [static]
```

Definition at line 303 of file [dhcp.py](#).

7.5.3.2 broadcast_address [2/2]

```
dhcp.DHCPServerConfiguration.broadcast_address
```

Definition at line 338 of file [dhcp.py](#).

7.5.3.3 debug

```
dhcp.DHCPServerConfiguration.debug = lambda *args, **kw: None [static]
```

Definition at line 312 of file [dhcp.py](#).

7.5.3.4 dhcp_acknowledge_after_seconds

```
int dhcp.DHCPServerConfiguration.dhcp_acknowledge_after_seconds = 10 [static]
```

Definition at line 299 of file [dhcp.py](#).

7.5.3.5 dhcp_offer_after_seconds

```
int dhcp.DHCPServerConfiguration.dhcp_offer_after_seconds = 10 [static]
```

Definition at line 298 of file [dhcp.py](#).

7.5.3.6 domain_name_server

```
dhcp.DHCPServerConfiguration.domain_name_server = None [static]
```

Definition at line 308 of file [dhcp.py](#).

7.5.3.7 host_file

```
string dhcp.DHCPServerConfiguration.host_file = 'hosts.csv' [static]
```

Definition at line 310 of file [dhcp.py](#).

7.5.3.8 ip_address_lease_time

```
int dhcp.DHCPServerConfiguration.ip_address_lease_time = 300 [static]
```

Definition at line 307 of file [dhcp.py](#).

7.5.3.9 length_of_transaction

```
int dhcp.DHCPServerConfiguration.length_of_transaction = 40 [static]
```

Definition at line 300 of file [dhcp.py](#).

7.5.3.10 network [1/2]

```
string dhcp.DHCPServerConfiguration.network = '192.168.173.0' [static]
```

Definition at line 302 of file [dhcp.py](#).

7.5.3.11 network [2/2]

```
dhcp.DHCPServerConfiguration.network
```

Definition at line 337 of file [dhcp.py](#).

7.5.3.12 router

```
dhcp.DHCPServerConfiguration.router = None [static]
```

Definition at line 305 of file [dhcp.py](#).

7.5.3.13 subnet_mask

```
string dhcp.DHCPServerConfiguration.subnet_mask = '255.255.255.0' [static]
```

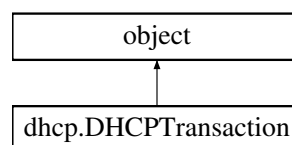
Definition at line 304 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.6 dhcp.DHCPTTransaction Class Reference

Inheritance diagram for dhcp.DHCPTTransaction:



Public Member Functions

- def `__init__` (self, [server](#))
- def `is_done` (self)
- def `close` (self)
- def `receive` (self, packet)
- def `received_dhcp_discover` (self, discovery)
- def `send_offer` (self, discovery)
- def `received_dhcp_request` (self, request)
- def `acknowledge` (self, request)
- def `received_dhcp_inform` (self, inform)

Public Attributes

- [server](#)
- [configuration](#)
- [packets](#)
- [done_time](#)
- [done](#)
- [do_after](#)
- [debug](#)

7.6.1 Detailed Description

Class representing DHCP Transaction

Definition at line [194](#) of file [dhcp.py](#).

7.6.2 Constructor & Destructor Documentation

7.6.2.1 `__init__()`

```
def dhcp.DHCPTransaction.__init__ (  
    self,  
    server )
```

Constructor of new transaction

Definition at line [197](#) of file [dhcp.py](#).

7.6.3 Member Function Documentation

7.6.3.1 acknowledge()

```
def dhcp.DHCPTransaction.acknowledge (
    self,
    request )
```

Method used to handle DHCP Acknowledge packet

Definition at line 270 of file [dhcp.py](#).

7.6.3.2 close()

```
def dhcp.DHCPTransaction.close (
    self )
```

Close transaction

Definition at line 213 of file [dhcp.py](#).

7.6.3.3 is_done()

```
def dhcp.DHCPTransaction.is_done (
    self )
```

Check if transaction is done

Definition at line 208 of file [dhcp.py](#).

7.6.3.4 receive()

```
def dhcp.DHCPTransaction.receive (
    self,
    packet )
```

Receive DHCP UDP packet check it's type and call a proper callback

Definition at line 218 of file [dhcp.py](#).

7.6.3.5 received_dhcp_discover()

```
def dhcp.DHCPTransaction.received_dhcp_discover (
    self,
    discovery )
```

Method used to handle DHCP Discover packet

Definition at line [234](#) of file [dhcp.py](#).

7.6.3.6 received_dhcp_inform()

```
def dhcp.DHCPTransaction.received_dhcp_inform (
    self,
    inform )
```

Method used to handle DHCP Inform packet

Definition at line [288](#) of file [dhcp.py](#).

7.6.3.7 received_dhcp_request()

```
def dhcp.DHCPTransaction.received_dhcp_request (
    self,
    request )
```

Method used to handle DHCP Request packet

Definition at line [261](#) of file [dhcp.py](#).

7.6.3.8 send_offer()

```
def dhcp.DHCPTransaction.send_offer (
    self,
    discovery )
```

Method used to send DHCP offer packet

Definition at line [241](#) of file [dhcp.py](#).

7.6.4 Member Data Documentation

7.6.4.1 configuration

`dhcp.DHCPTransaction.configuration`

Definition at line 201 of file [dhcp.py](#).

7.6.4.2 debug

`dhcp.DHCPTransaction.debug`

Definition at line 206 of file [dhcp.py](#).

7.6.4.3 do_after

`dhcp.DHCPTransaction.do_after`

Definition at line 205 of file [dhcp.py](#).

7.6.4.4 done

`dhcp.DHCPTransaction.done`

Definition at line 204 of file [dhcp.py](#).

7.6.4.5 done_time

`dhcp.DHCPTransaction.done_time`

Definition at line 203 of file [dhcp.py](#).

7.6.4.6 packets

`dhcp.DHCPTransaction.packets`

Definition at line 202 of file [dhcp.py](#).

7.6.4.7 server

`dhcp.DHCPTransaction.server`

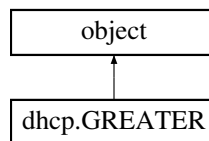
Definition at line 200 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.7 dhcp.GREATER Class Reference

Inheritance diagram for `dhcp.GREATER`:



Public Member Functions

- `def __init__(self, value)`
- `def __eq__(self, other)`

Public Attributes

- [value](#)

7.7.1 Detailed Description

Comparator class

Definition at line 371 of file [dhcp.py](#).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 `__init__()`

```
def dhcp.GREATER.__init__ (
    self,
    value )
```

Definition at line 374 of file [dhcp.py](#).

7.7.3 Member Function Documentation

7.7.3.1 `__eq__()`

```
def dhcp.GREATER.__eq__ (
    self,
    other )
```

Definition at line 376 of file [dhcp.py](#).

7.7.4 Member Data Documentation

7.7.4.1 `value`

```
dhcp.GREATER.value
```

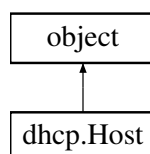
Definition at line 375 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.8 dhcp.Host Class Reference

Inheritance diagram for `dhcp.Host`:



Public Member Functions

- def `__init__` (self, `mac`, `ip`, `hostname`, `last_used`)
- def `from_tuple` (cls, line)
- def `from_packet` (cls, packet)
- def `to_tuple` (self)
- def `to_pattern` (self)
- def `__hash__` (self)
- def `__eq__` (self, other)
- def `has_valid_ip` (self)

Static Public Member Functions

- def `get_pattern` (`mac=ALL`, `ip=ALL`, `hostname=ALL`, `last_used=ALL`)

Public Attributes

- `mac`
- `ip`
- `hostname`
- `last_used`

7.8.1 Detailed Description

Class representing host with MAC address, IP, hostname if available and last used timestamp

Definition at line 443 of file `dhcp.py`.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `__init__()`

```
def dhcp.Host.__init__ (  
    self,  
    mac,  
    ip,  
    hostname,  
    last_used )
```

Definition at line 446 of file `dhcp.py`.

7.8.3 Member Function Documentation

7.8.3.1 `__eq__()`

```
def dhcp.Host.__eq__ (
    self,
    other )
```

Definition at line 482 of file [dhcp.py](#).

7.8.3.2 `__hash__()`

```
def dhcp.Host.__hash__ (
    self )
```

Definition at line 479 of file [dhcp.py](#).

7.8.3.3 `from_packet()`

```
def dhcp.Host.from_packet (
    cls,
    packet )
```

Definition at line 459 of file [dhcp.py](#).

7.8.3.4 `from_tuple()`

```
def dhcp.Host.from_tuple (
    cls,
    line )
```

Definition at line 453 of file [dhcp.py](#).

7.8.3.5 `get_pattern()`

```
def dhcp.Host.get_pattern (
    mac = ALL,
    ip = ALL,
    hostname = ALL,
    last_used = ALL ) [static]
```

Definition at line 466 of file [dhcp.py](#).

7.8.3.6 has_valid_ip()

```
def dhcp.Host.has_valid_ip (  
    self )
```

Check if host has valid IP address

Definition at line [485](#) of file [dhcp.py](#).

7.8.3.7 to_pattern()

```
def dhcp.Host.to_pattern (  
    self )
```

Convert host to pattern

Definition at line [474](#) of file [dhcp.py](#).

7.8.3.8 to_tuple()

```
def dhcp.Host.to_tuple (  
    self )
```

Convert host to tuple

Definition at line [469](#) of file [dhcp.py](#).

7.8.4 Member Data Documentation

7.8.4.1 hostname

```
dhcp.Host.hostname
```

Definition at line [449](#) of file [dhcp.py](#).

7.8.4.2 ip

`dhcp.Host.ip`

Definition at line 448 of file [dhcp.py](#).

7.8.4.3 last_used

`dhcp.Host.last_used`

Definition at line 450 of file [dhcp.py](#).

7.8.4.4 mac

`dhcp.Host.mac`

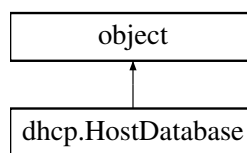
Definition at line 447 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.9 dhcp.HostDatabase Class Reference

Inheritance diagram for `dhcp.HostDatabase`:



Public Member Functions

- `def __init__(self, file_name)`
- `def get(self, **kw)`
- `def add(self, host)`
- `def delete(self, host=None, **kw)`
- `def all(self)`
- `def replace(self, host)`

Public Attributes

- [db](#)

7.9.1 Detailed Description

Definition at line [490](#) of file [dhcp.py](#).

7.9.2 Constructor & Destructor Documentation

7.9.2.1 `__init__()`

```
def dhcp.HostDatabase.__init__ (
    self,
    file_name )
```

Definition at line [491](#) of file [dhcp.py](#).

7.9.3 Member Function Documentation

7.9.3.1 `add()`

```
def dhcp.HostDatabase.add (
    self,
    host )
```

Definition at line [498](#) of file [dhcp.py](#).

7.9.3.2 `all()`

```
def dhcp.HostDatabase.all (
    self )
```

Definition at line [508](#) of file [dhcp.py](#).

7.9.3.3 `delete()`

```
def dhcp.HostDatabase.delete (
    self,
    host = None,
    ** kw )
```

Definition at line [501](#) of file [dhcp.py](#).

7.9.3.4 get()

```
def dhcp.HostDatabase.get (
    self,
    ** kw )
```

Definition at line 494 of file [dhcp.py](#).

7.9.3.5 replace()

```
def dhcp.HostDatabase.replace (
    self,
    host )
```

Definition at line 511 of file [dhcp.py](#).

7.9.4 Member Data Documentation

7.9.4.1 db

```
dhcp.HostDatabase.db
```

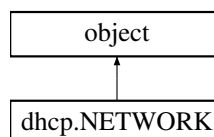
Definition at line 492 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.10 dhcp.NETWORK Class Reference

Inheritance diagram for dhcp.NETWORK:



Public Member Functions

- `def __init__(self, network, subnet_mask)`
- `def __eq__(self, other)`

Public Attributes

- [subnet_mask](#)
- [network](#)

7.10.1 Detailed Description

Comparator class to check if address within same network

Definition at line [379](#) of file [dhcp.py](#).

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `__init__()`

```
def dhcp.NETWORK.__init__ (
    self,
    network,
    subnet_mask )
```

Definition at line [382](#) of file [dhcp.py](#).

7.10.3 Member Function Documentation

7.10.3.1 `__eq__()`

```
def dhcp.NETWORK.__eq__ (
    self,
    other )
```

Definition at line [385](#) of file [dhcp.py](#).

7.10.4 Member Data Documentation

7.10.4.1 `network`

`dhcp.NETWORK.network`

Definition at line [384](#) of file [dhcp.py](#).

7.10.4.2 subnet_mask

`dhcp.NETWORK.subnet_mask`

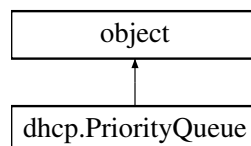
Definition at line 383 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.11 dhcp.PriorityQueue Class Reference

Inheritance diagram for `dhcp.PriorityQueue`:



Public Member Functions

- `def __init__(self)`
- `def put(self, item)`
- `def get(self)`
- `def qsize(self)`

7.11.1 Detailed Description

This class contains `Heapq` for more information:
<https://docs.python.org/3/library/heapq.html>

Definition at line 64 of file [dhcp.py](#).

7.11.2 Constructor & Destructor Documentation

7.11.2.1 __init__()

```
def dhcp.PriorityQueue.__init__(  
    self )
```

Definition at line 68 of file [dhcp.py](#).

7.11.3 Member Function Documentation

7.11.3.1 `get()`

```
def dhcp.PriorityQueue.get (
    self )
```

Definition at line 76 of file [dhcp.py](#).

7.11.3.2 `put()`

```
def dhcp.PriorityQueue.put (
    self,
    item )
```

Definition at line 72 of file [dhcp.py](#).

7.11.3.3 `qsize()`

```
def dhcp.PriorityQueue.qsize (
    self )
```

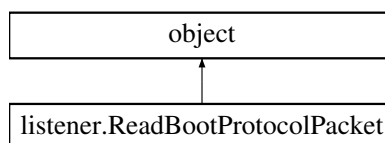
Definition at line 79 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.12 `listener.ReadBootProtocolPacket` Class Reference

Inheritance diagram for `listener.ReadBootProtocolPacket`:



Public Member Functions

- def [__init__](#) (self, [data](#), [address](#)=('0.0.0.0', 0))
- def [__getitem__](#) (self, key)
- def [__contains__](#) (self, key)
- def [formatted_named_options](#) (self)
- def [__str__](#) (self)
- def [__gt__](#) (self, other)

Public Attributes

- [data](#)
- [address](#)
- [host](#)
- [port](#)
- [message_type](#)
- [OP](#)
- [hardware_type](#)
- [HTYPE](#)
- [hardware_address_length](#)
- [HLEN](#)
- [hops](#)
- [HOPS](#)
- [XID](#)
- [transaction_id](#)
- [seconds_elapsed](#)
- [SECS](#)
- [bootp_flags](#)
- [FLAGS](#)
- [client_ip_address](#)
- [CIADDR](#)
- [your_ip_address](#)
- [YIADDR](#)
- [next_server_ip_address](#)
- [SIADDR](#)
- [relay_agent_ip_address](#)
- [GIADDR](#)
- [client_mac_address](#)
- [CHADDR](#)
- [magic_cookie](#)
- [options](#)
- [named_options](#)

7.12.1 Detailed Description

Definition at line 144 of file [listener.py](#).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 `__init__()`

```
def listener.ReadBootProtocolPacket.__init__ (
    self,
    data,
    address = ('0.0.0.0', 0) )
```

Definition at line 152 of file [listener.py](#).

7.12.3 Member Function Documentation

7.12.3.1 `__contains__()`

```
def listener.ReadBootProtocolPacket.__contains__ (
    self,
    key )
```

Definition at line 205 of file [listener.py](#).

7.12.3.2 `__getitem__()`

```
def listener.ReadBootProtocolPacket.__getitem__ (
    self,
    key )
```

Definition at line 201 of file [listener.py](#).

7.12.3.3 `__gt__()`

```
def listener.ReadBootProtocolPacket.__gt__ (
    self,
    other )
```

Definition at line 221 of file [listener.py](#).

7.12.3.4 `__str__()`

```
def listener.ReadBootProtocolPacket.__str__ (
    self )
```

Definition at line 212 of file [listener.py](#).

7.12.3.5 formatted_named_options()

```
def listener.ReadBootProtocolPacket.formatted_named_options (
    self )
```

Definition at line 209 of file [listener.py](#).

7.12.4 Member Data Documentation

7.12.4.1 address

```
listener.ReadBootProtocolPacket.address
```

Definition at line 154 of file [listener.py](#).

7.12.4.2 bootp_flags

```
listener.ReadBootProtocolPacket.bootp_flags
```

Definition at line 168 of file [listener.py](#).

7.12.4.3 CHADDR

```
listener.ReadBootProtocolPacket.CHADDR
```

Definition at line 175 of file [listener.py](#).

7.12.4.4 CIADDR

```
listener.ReadBootProtocolPacket.CIADDR
```

Definition at line 170 of file [listener.py](#).

7.12.4.5 client_ip_address

```
listener.ReadBootProtocolPacket.client_ip_address
```

Definition at line 170 of file [listener.py](#).

7.12.4.6 client_mac_address

`listener.ReadBootProtocolPacket.client_mac_address`

Definition at line 175 of file [listener.py](#).

7.12.4.7 data

`listener.ReadBootProtocolPacket.data`

Definition at line 153 of file [listener.py](#).

7.12.4.8 FLAGS

`listener.ReadBootProtocolPacket.FLAGS`

Definition at line 168 of file [listener.py](#).

7.12.4.9 GIADDR

`listener.ReadBootProtocolPacket.GIADDR`

Definition at line 173 of file [listener.py](#).

7.12.4.10 hardware_address_length

`listener.ReadBootProtocolPacket.hardware_address_length`

Definition at line 162 of file [listener.py](#).

7.12.4.11 hardware_type

`listener.ReadBootProtocolPacket.hardware_type`

Definition at line 161 of file [listener.py](#).

7.12.4.12 HLEN

`listener.ReadBootProtocolPacket.HLEN`

Definition at line 162 of file [listener.py](#).

7.12.4.13 hops

`listener.ReadBootProtocolPacket.hops`

Definition at line 163 of file [listener.py](#).

7.12.4.14 HOPS

`listener.ReadBootProtocolPacket.HOPS`

Definition at line 163 of file [listener.py](#).

7.12.4.15 host

`listener.ReadBootProtocolPacket.host`

Definition at line 155 of file [listener.py](#).

7.12.4.16 HTYPE

`listener.ReadBootProtocolPacket.HTYPE`

Definition at line 161 of file [listener.py](#).

7.12.4.17 magic_cookie

`listener.ReadBootProtocolPacket.magic_cookie`

Definition at line 177 of file [listener.py](#).

7.12.4.18 message_type

`listener.ReadBootProtocolPacket.message_type`

Definition at line 160 of file [listener.py](#).

7.12.4.19 named_options

`listener.ReadBootProtocolPacket.named_options`

Definition at line 179 of file [listener.py](#).

7.12.4.20 next_server_ip_address

`listener.ReadBootProtocolPacket.next_server_ip_address`

Definition at line 172 of file [listener.py](#).

7.12.4.21 OP

`listener.ReadBootProtocolPacket.OP`

Definition at line 160 of file [listener.py](#).

7.12.4.22 options

`listener.ReadBootProtocolPacket.options`

Definition at line 178 of file [listener.py](#).

7.12.4.23 port

`listener.ReadBootProtocolPacket.port`

Definition at line 156 of file [listener.py](#).

7.12.4.24 relay_agent_ip_address

`listener.ReadBootProtocolPacket.relay_agent_ip_address`

Definition at line 173 of file [listener.py](#).

7.12.4.25 seconds_elapsed

`listener.ReadBootProtocolPacket.seconds_elapsed`

Definition at line 167 of file [listener.py](#).

7.12.4.26 SECS

`listener.ReadBootProtocolPacket.SECS`

Definition at line 167 of file [listener.py](#).

7.12.4.27 SIADDR

`listener.ReadBootProtocolPacket.SIADDR`

Definition at line 172 of file [listener.py](#).

7.12.4.28 transaction_id

`listener.ReadBootProtocolPacket.transaction_id`

Definition at line 165 of file [listener.py](#).

7.12.4.29 XID

`listener.ReadBootProtocolPacket.XID`

Definition at line 165 of file [listener.py](#).

7.12.4.30 YIADDR

```
listener.ReadBootProtocolPacket.YIADDR
```

Definition at line 171 of file [listener.py](#).

7.12.4.31 your_ip_address

```
listener.ReadBootProtocolPacket.your_ip_address
```

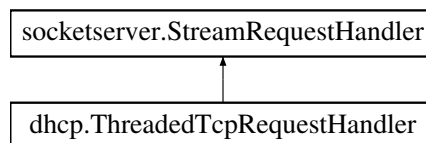
Definition at line 171 of file [listener.py](#).

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

- [listener.py](#)

7.13 dhcp.ThreadedTcpRequestHandler Class Reference

Inheritance diagram for dhcp.ThreadedTcpRequestHandler:



Public Member Functions

- def [handle](#) (self)

7.13.1 Detailed Description

```
Control socket client connection handler
```

Definition at line 660 of file [dhcp.py](#).

7.13.2 Member Function Documentation

7.13.2.1 handle()

```
def dhcp.ThreadedTcpRequestHandler.handle (
    self )
```

Method used to handle client connection parsing commands and giving response to them

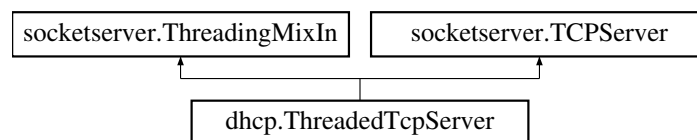
Definition at line 663 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.14 dhcp.ThreadedTcpServer Class Reference

Inheritance diagram for dhcp.ThreadedTcpServer:



Public Member Functions

- def [setEvents](#) (self, data)
- def [setHosts](#) (self, data)
- def [setConfiguration](#) (self, data)

Public Attributes

- [events](#)
- [hosts](#)
- [configuration](#)

7.14.1 Detailed Description

DHCP server control interface TCP server

Definition at line 694 of file [dhcp.py](#).

7.14.2 Member Function Documentation

7.14.2.1 setConfiguration()

```
def dhcp.ThreadedTcpServer.setConfiguration (
    self,
    data )
```

Set DHCP UDP global configuration reference

Definition at line 707 of file [dhcp.py](#).

7.14.2.2 setEvents()

```
def dhcp.ThreadedTcpServer.setEvents (
    self,
    data )
```

Set DHCP events dictionary reference

Definition at line 697 of file [dhcp.py](#).

7.14.2.3 setHosts()

```
def dhcp.ThreadedTcpServer.setHosts (
    self,
    data )
```

Set DHCP host database with active leases reference

Definition at line 702 of file [dhcp.py](#).

7.14.3 Member Data Documentation

7.14.3.1 configuration

```
dhcp.ThreadedTcpServer.configuration
```

Definition at line 710 of file [dhcp.py](#).

7.14.3.2 events

`dhcp.ThreadedTcpServer.events`

Definition at line 700 of file [dhcp.py](#).

7.14.3.3 hosts

`dhcp.ThreadedTcpServer.hosts`

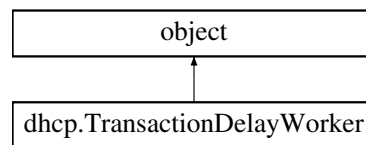
Definition at line 705 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.15 dhcp.TransactionDelayWorker Class Reference

Inheritance diagram for `dhcp.TransactionDelayWorker`:



Public Member Functions

- `def __init__ (self)`
- `def do_after (self, seconds, func, args=(), kw={})`
- `def close (self)`

Public Attributes

- [closed](#)
- [queue](#)
- [thread](#)

7.15.1 Detailed Description

Class used to delay response to DHCP client

Definition at line 20 of file [dhcp.py](#).

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `__init__()`

```
def dhcp.TransactionDelayWorker.__init__ (
    self )
```

class constructor internally using priority queue where priority is time

Definition at line 23 of file [dhcp.py](#).

7.15.3 Member Function Documentation

7.15.3.1 `close()`

```
def dhcp.TransactionDelayWorker.close (
    self )
```

Method used to stop worker

Definition at line 54 of file [dhcp.py](#).

7.15.3.2 `do_after()`

```
def dhcp.TransactionDelayWorker.do_after (
    self,
    seconds,
    func,
    args = (),
    kw = {} )
```

Add to queue function which should be called after certain time specified by seconds, args, kw are arguments

Definition at line 48 of file [dhcp.py](#).

7.15.4 Member Data Documentation

7.15.4.1 closed

`dhcp.TransactionDelayWorker.closed`

Definition at line 26 of file [dhcp.py](#).

7.15.4.2 queue

`dhcp.TransactionDelayWorker.queue`

Definition at line 27 of file [dhcp.py](#).

7.15.4.3 thread

`dhcp.TransactionDelayWorker.thread`

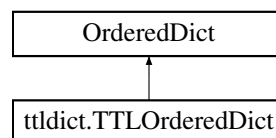
Definition at line 28 of file [dhcp.py](#).

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

- [dhcp.py](#)

7.16 ttlDict.TTLOrderedDict Class Reference

Inheritance diagram for `ttlDict.TTLOrderedDict`:



Public Member Functions

- `def __init__(self, default_ttl, *args, **kwargs)`
- `def __repr__(self)`
- `def __len__(self)`
- `def set_ttl(self, key, ttl, now=None)`
- `def get_ttl(self, key, now=None)`
- `def expire_at(self, key, timestamp)`
- `def is_expired(self, key, now=None)`
- `def __iter__(self)`
- `def __setitem__(self, key, value)`
- `def __delitem__(self, key)`
- `def __getitem__(self, key)`
- `def keys(self)`
- `def items(self)`
- `def values(self)`
- `def get(self, key, default=None)`

7.16.1 Detailed Description

OrderedDict with TTL
Extra args and kwargs are passed to initial `.update()` call

Definition at line 8 of file [ttl dict.py](#).

7.16.2 Constructor & Destructor Documentation

7.16.2.1 `__init__()`

```
def ttl dict.TTLOrderedDict.__init__ (
    self,
    default_ttl,
    * args,
    ** kwargs )
```

Be warned, if you use this with Python versions earlier than 3.6
when passing `**kwargs` order is not preseverd.

Definition at line 13 of file [ttl dict.py](#).

7.16.3 Member Function Documentation

7.16.3.1 `__delitem__()`

```
def ttl dict.TTLOrderedDict.__delitem__ (
    self,
    key )
```

Definition at line 89 of file [ttl dict.py](#).

7.16.3.2 `__getitem__()`

```
def ttl dict.TTLOrderedDict.__getitem__ (
    self,
    key )
```

Definition at line 93 of file [ttl dict.py](#).

7.16.3.3 `__iter__()`

```
def ttldict.TTLOrderedDict.__iter__ (
    self )
```

Yield only non expired keys, without purging the expired ones

Definition at line 72 of file [ttldict.py](#).

7.16.3.4 `__len__()`

```
def ttldict.TTLOrderedDict.__len__ (
    self )
```

Definition at line 28 of file [ttldict.py](#).

7.16.3.5 `__repr__()`

```
def ttldict.TTLOrderedDict.__repr__ (
    self )
```

Definition at line 24 of file [ttldict.py](#).

7.16.3.6 `__setitem__()`

```
def ttldict.TTLOrderedDict.__setitem__ (
    self,
    key,
    value )
```

Definition at line 81 of file [ttldict.py](#).

7.16.3.7 `expire_at()`

```
def ttldict.TTLOrderedDict.expire_at (
    self,
    key,
    timestamp )
```

Set the key expire timestamp

Definition at line 49 of file [ttldict.py](#).

7.16.3.8 `get()`

```
def ttldict.TTLOrderedDict.get (
    self,
    key,
    default = None )
```

Definition at line 118 of file [ttldict.py](#).

7.16.3.9 `get_ttl()`

```
def ttldict.TTLOrderedDict.get_ttl (
    self,
    key,
    now = None )
```

Return remaining TTL for a key

Definition at line 41 of file [ttldict.py](#).

7.16.3.10 `is_expired()`

```
def ttldict.TTLOrderedDict.is_expired (
    self,
    key,
    now = None )
```

Check if key has expired, and return it if so

Definition at line 55 of file [ttldict.py](#).

7.16.3.11 `items()`

```
def ttldict.TTLOrderedDict.items (
    self )
```

Definition at line 106 of file [ttldict.py](#).

7.16.3.12 keys()

```
def ttldict.TTLOrderedDict.keys (
    self )
```

Definition at line 101 of file [ttldict.py](#).

7.16.3.13 set_ttl()

```
def ttldict.TTLOrderedDict.set_ttl (
    self,
    key,
    ttl,
    now = None )
```

Set TTL for the given key

Definition at line 33 of file [ttldict.py](#).

7.16.3.14 values()

```
def ttldict.TTLOrderedDict.values (
    self )
```

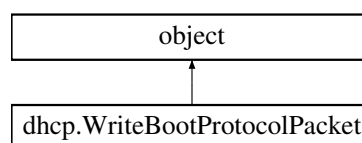
Definition at line 112 of file [ttldict.py](#).

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

- [ttldict.py](#)

7.17 dhcp.WriteBootProtocolPacket Class Reference

Inheritance diagram for dhcp.WriteBootProtocolPacket:



Public Member Functions

- def `__init__` (self, [configuration](#))
- def [to_bytes](#) (self)
- def [get_option](#) (self, option)
- def [options](#) (self)
- def `__str__` (self)

Static Public Attributes

- int [message_type](#) = 2
- int [hardware_type](#) = 1
- int [hardware_address_length](#) = 6
- int [hops](#) = 0
- [transaction_id](#) = None
- int [seconds_elapsed](#) = 0
- int [bootp_flags](#) = 0
- string [client_ip_address](#) = '0.0.0.0'
- string [your_ip_address](#) = '0.0.0.0'
- string [next_server_ip_address](#) = '0.0.0.0'
- string [relay_agent_ip_address](#) = '0.0.0.0'
- [client_mac_address](#) = None
- string [magic_cookie](#) = '99.130.83.99'
- list [parameter_order](#) = []

7.17.1 Detailed Description

DHCP protocol datagram serializer

This class serializes UDP DHCP packet, instance is constructed using global configuration from which dhcp options are copied

Definition at line 82 of file [dhcp.py](#).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 `__init__()`

```
def dhcp.WriteBootProtocolPacket.__init__ (
    self,
    configuration )
```

Create new packet instance and search for options set in configuration and copy them to packet

Definition at line 107 of file [dhcp.py](#).

7.17.3 Member Function Documentation

7.17.3.1 `__str__()`

```
def dhcp.WriteBootProtocolPacket.__str__ (
    self )
```

Serialize UDP DHCP response packet to bytes

Definition at line 188 of file [dhcp.py](#).

7.17.3.2 `get_option()`

```
def dhcp.WriteBootProtocolPacket.get_option (
    self,
    option )
```

Get DHCP UDP response packet option value

Definition at line 152 of file [dhcp.py](#).

7.17.3.3 `options()`

```
def dhcp.WriteBootProtocolPacket.options (
    self )
```

Get DHCP UDP response packet option value

Definition at line 167 of file [dhcp.py](#).

7.17.3.4 `to_bytes()`

```
def dhcp.WriteBootProtocolPacket.to_bytes (
    self )
```

Serialize UDP DHCP response packet to bytes

Definition at line 119 of file [dhcp.py](#).

7.17.4 Member Data Documentation

7.17.4.1 bootp_flags

```
int dhcp.WriteBootProtocolPacket.bootp_flags = 0 [static]
```

Definition at line 95 of file [dhcp.py](#).

7.17.4.2 client_ip_address

```
string dhcp.WriteBootProtocolPacket.client_ip_address = '0.0.0.0' [static]
```

Definition at line 97 of file [dhcp.py](#).

7.17.4.3 client_mac_address

```
dhcp.WriteBootProtocolPacket.client_mac_address = None [static]
```

Definition at line 102 of file [dhcp.py](#).

7.17.4.4 hardware_address_length

```
int dhcp.WriteBootProtocolPacket.hardware_address_length = 6 [static]
```

Definition at line 89 of file [dhcp.py](#).

7.17.4.5 hardware_type

```
int dhcp.WriteBootProtocolPacket.hardware_type = 1 [static]
```

Definition at line 88 of file [dhcp.py](#).

7.17.4.6 hops

```
int dhcp.WriteBootProtocolPacket.hops = 0 [static]
```

Definition at line 90 of file [dhcp.py](#).

7.17.4.7 magic_cookie

```
string dhcp.WriteBootProtocolPacket.magic_cookie = '99.130.83.99' [static]
```

Definition at line 103 of file [dhcp.py](#).

7.17.4.8 message_type

```
int dhcp.WriteBootProtocolPacket.message_type = 2 [static]
```

Definition at line 87 of file [dhcp.py](#).

7.17.4.9 next_server_ip_address

```
string dhcp.WriteBootProtocolPacket.next_server_ip_address = '0.0.0.0' [static]
```

Definition at line 99 of file [dhcp.py](#).

7.17.4.10 parameter_order

```
list dhcp.WriteBootProtocolPacket.parameter_order = [] [static]
```

Definition at line 105 of file [dhcp.py](#).

7.17.4.11 relay_agent_ip_address

```
string dhcp.WriteBootProtocolPacket.relay_agent_ip_address = '0.0.0.0' [static]
```

Definition at line 100 of file [dhcp.py](#).

7.17.4.12 seconds_elapsed

```
int dhcp.WriteBootProtocolPacket.seconds_elapsed = 0 [static]
```

Definition at line 94 of file [dhcp.py](#).

7.17.4.13 transaction_id

```
dhcp.WriteBootProtocolPacket.transaction_id = None [static]
```

Definition at line 92 of file [dhcp.py](#).

7.17.4.14 your_ip_address

```
string dhcp.WriteBootProtocolPacket.your_ip_address = '0.0.0.0' [static]
```

Definition at line 98 of file [dhcp.py](#).

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

- [dhcp.py](#)

Chapter 8

File Documentation

8.1 dhcp.py File Reference

Classes

- class [dhcp.TransactionDelayWorker](#)
- class [dhcp.PriorityQueue](#)
- class [dhcp.WriteBootProtocolPacket](#)
- class [dhcp.DHCPTransaction](#)
- class [dhcp.DHCPServerConfiguration](#)
- class [dhcp.ALL](#)
- class [dhcp.GREATER](#)
- class [dhcp.NETWORK](#)
- class [dhcp.CASEINSENSITIVE](#)
- class [dhcp.CSVDatabase](#)
- class [dhcp.Host](#)
- class [dhcp.HostDatabase](#)
- class [dhcp.DHCPServer](#)
- class [dhcp.ThreadedTcpRequestHandler](#)
- class [dhcp.ThreadedTcpServer](#)

Namespaces

- namespace [dhcp](#)

Functions

- def [dhcp.get_host_ip_addresses](#) ()
- def [dhcp.ip_addresses](#) (network, subnet_mask)
- def [dhcp.sorted_hosts](#) (hosts)
- def [dhcp.debug_msg](#) (msg, type)

Variables

- `dhcp.ALL` = `ALL()`
- `dhcp.messages` = `TTLOrderedDict(default_ttl=86400)`
- `dhcp.configuration` = `DHCPServerConfiguration()`
- `dhcp.debug`
- `dhcp.ip_address_lease_time`
- `dhcp.server` = `DHCPServer(configuration)`

8.2 dhcp.py

[Go to the documentation of this file.](#)

```
00001 #!/usr/bin/env python3
00002
00003 import time
00004 import threading
00005 import struct
00006 import queue
00007 import collections
00008 import traceback
00009 import random
00010 import socket
00011 import heapq
00012 import sys
00013 from os.path import exists
00014 import re
00015 from ttldict import TTLOrderedDict
00016 import socketserver
00017 from listener import *
00018
00019
00020 class TransactionDelayWorker(object):
00021     """Class used to delay response to DHCP client
00022     """
00023     def __init__(self):
00024         """class constructor internally using priority queue where priority is time
00025         """
00026         self.closed = False
00027         self.queue = PriorityQueue()
00028         self.thread = threading.Thread(target = self._delay_response_thread)
00029         self.thread.start()
00030
00031     def _delay_response_thread(self):
00032         """thread worker
00033         """
00034         while not self.closed:
00035             if self.closed:
00036                 break
00037             if self.queue.qsize() > 0:
00038                 p = self.queue.get()
00039                 t, func, args, kw = p
00040                 now = time.time()
00041                 if now < t:
00042                     time.sleep(0.01)
00043                     self.queue.put(p)
00044                 else:
00045                     func(*args, **kw)
00046
00047
00048     def do_after(self, seconds, func, args = (), kw = {}):
00049         """Add to queue function which should be called after certain time
00050         specified by seconds, args, kw are arguments
00051         """
00052         self.queue.put((time.time() + seconds, func, args, kw))
00053
00054     def close(self):
00055         """Method used to stop worker
00056         """
00057         self.closed = True
00058
00059 def get_host_ip_addresses():
00060     """Get IP address of current host.
00061     """
00062     return gethostbyname_ex(gethostname())[2]
00063
00064 class PriorityQueue(object):
00065     """This class contains Heapq for more information:
```

```

00066     https://docs.python.org/3/library/heapq.html
00067     """
00068     def __init__(self):
00069         self._queue = []
00070         self._index = 0
00071
00072     def put(self, item):
00073         heapq.heappush(self._queue, (self._index, item))
00074         self._index += 1
00075
00076     def get(self):
00077         return heapq.heappop(self._queue)[-1]
00078
00079     def qsize(self):
00080         return len(self._queue)
00081
00082 class WriteBootProtocolPacket(object):
00083     """DHCP protocol datagram serializer
00084     This class serializes UDP DHCP packet, instance is constructed using global
00085     configuration from which dhcp options are copied
00086     """
00087     message_type = 2 # 1 for client -> server 2 for server -> client
00088     hardware_type = 1
00089     hardware_address_length = 6
00090     hops = 0
00091
00092     transaction_id = None
00093
00094     seconds_elapsed = 0
00095     bootp_flags = 0 # unicast
00096
00097     client_ip_address = '0.0.0.0'
00098     your_ip_address = '0.0.0.0'
00099     next_server_ip_address = '0.0.0.0'
00100     relay_agent_ip_address = '0.0.0.0'
00101
00102     client_mac_address = None
00103     magic_cookie = '99.130.83.99'
00104
00105     parameter_order = []
00106
00107     def __init__(self, configuration):
00108         """Create new packet instance and search for options set in configuration
00109         and copy them tgo packet
00110         """
00111         for i in range(256):
00112             names = ['option_{}'.format(i)]
00113             if i < len(options) and hasattr(configuration, options[i][0]):
00114                 names.append(options[i][0])
00115             for name in names:
00116                 if hasattr(configuration, name):
00117                     setattr(self, name, getattr(configuration, name))
00118
00119     def to_bytes(self):
00120         """Serialize UDP DHCP response packet to bytes
00121         """
00122         result = bytearray(236)
00123
00124         result[0] = self.message_type
00125         result[1] = self.hardware_type
00126         result[2] = self.hardware_address_length
00127         result[3] = self.hops
00128
00129         result[4:8] = struct.pack('>I', self.transaction_id)
00130
00131         result[8:10] = shortpack(self.seconds_elapsed)
00132         result[10:12] = shortpack(self.bootp_flags)
00133
00134         result[12:16] = inet_aton(self.client_ip_address)
00135         result[16:20] = inet_aton(self.your_ip_address)
00136         result[20:24] = inet_aton(self.next_server_ip_address)
00137         result[24:28] = inet_aton(self.relay_agent_ip_address)
00138
00139         result[28:28 + self.hardware_address_length] = macpack(self.client_mac_address)
00140
00141         result += inet_aton(self.magic_cookie)
00142
00143         for option in self.options:
00144             value = self.get_option(option)
00145             #print(option, value)
00146             if value is None:
00147                 continue
00148             result += bytes([option, len(value)]) + value
00149         result += bytes([255])
00150         return bytes(result)
00151
00152     def get_option(self, option):

```

```

00153         """Get DHCP UDP response packet option value
00154         """
00155         if option < len(options) and hasattr(self, options[option][0]):
00156             value = getattr(self, options[option][0])
00157         elif hasattr(self, 'option_{}'.format(option)):
00158             value = getattr(self, 'option_{}'.format(option))
00159         else:
00160             return None
00161         function = options[option][2]
00162         if function and value is not None:
00163             value = function(value)
00164         return value
00165
00166     @property
00167     def options(self):
00168         """Get DHCP UDP response packet option value
00169         """
00170         done = list()
00171         # fulfill wishes
00172         for option in self.parameter_order:
00173             if option < len(options) and hasattr(self, options[option][0]) or hasattr(self,
'option_{}'.format(option)):
00174                 # this may break with the specification because we must try to fulfill the wishes
00175                 if option not in done:
00176                     done.append(option)
00177             # add my stuff
00178             for option, o in enumerate(options):
00179                 if o[0] and hasattr(self, o[0]):
00180                     if option not in done:
00181                         done.append(option)
00182             for option in range(256):
00183                 if hasattr(self, 'option_{}'.format(option)):
00184                     if option not in done:
00185                         done.append(option)
00186             return done
00187
00188     def __str__(self):
00189         """Serialize UDP DHCP response packet to bytes
00190         """
00191         return str(ReadBootProtocolPacket(self.to_bytes()))
00192
00193
00194 class DHCPTransaction(object):
00195     """Class representing DHCP Transaction
00196     """
00197     def __init__(self, server):
00198         """Constructor of new transaction
00199         """
00200         self.server = server
00201         self.configuration = server.configuration
00202         self.packets = []
00203         self.done_time = time.time() + self.configuration.length_of_transaction
00204         self.done = False
00205         self.do_after = self.server.delay_worker.do_after
00206         self.debug = debug
00207
00208     def is_done(self):
00209         """Check if transaction is done
00210         """
00211         return self.done or self.done_time < time.time()
00212
00213     def close(self):
00214         """Close transaction
00215         """
00216         self.done = True
00217
00218     def receive(self, packet):
00219         """Receive DHCP UDP packet check it's type and call a proper callback
00220         """
00221         # packet from client <-> packet.message_type == 1
00222         if packet.message_type == 1 and packet.dhcp_message_type == 'DHCPDISCOVER':
00223             self.do_after(self.configuration.dhcp_offer_after_seconds,
00224                             self.received_dhcp_discover, (packet,), )
00225         elif packet.message_type == 1 and packet.dhcp_message_type == 'DHCPREQUEST':
00226             self.do_after(self.configuration.dhcp_acknowledge_after_seconds,
00227                             self.received_dhcp_request, (packet,), )
00228         elif packet.message_type == 1 and packet.dhcp_message_type == 'DHCPINFORM':
00229             self.received_dhcp_inform(packet)
00230         else:
00231             return False
00232         return True
00233
00234     def received_dhcp_discover(self, discovery):
00235         """Method used to handle DHCP Discover packet
00236         """
00237         if self.is_done(): return
00238         self.configuration.debug('discover:\n {}'.format(str(discovery).replace('\n', '\n\t')))

```

```

00239         self.send_offer(discovery)
00240
00241     def send_offer(self, discovery):
00242         """Method used to send DHCP offer packet
00243         """
00244         # https://tools.ietf.org/html/rfc2131
00245         offer = WriteBootProtocolPacket(self.configuration)
00246         offer.parameter_order = discovery.parameter_request_list
00247         mac = discovery.client_mac_address
00248         ip = offer.your_ip_address = self.server.get_ip_address(discovery)
00249         # offer.client_ip_address =
00250         offer.transaction_id = discovery.transaction_id
00251         # offer.next_server_ip_address =
00252         offer.relay_agent_ip_address = discovery.relay_agent_ip_address
00253         offer.client_mac_address = mac
00254         offer.client_ip_address = discovery.client_ip_address or '0.0.0.0'
00255         offer.bootp_flags = discovery.bootp_flags
00256         offer.dhcp_message_type = 'DHCPOFFER'
00257         offer.client_identifier = mac
00258         self.configuration.debug('offer:\n {}'.format(str(offer).replace('\n', '\n\t')))
00259         self.server.broadcast(offer)
00260
00261     def received_dhcp_request(self, request):
00262         """Method used to handle DHCP Request packet
00263         """
00264         if self.is_done(): return
00265         self.configuration.debug('request:\n {}'.format(str(request).replace('\n', '\n\t')))
00266         self.server.client_has_chosen(request)
00267         self.acknowledge(request)
00268         self.close()
00269
00270     def acknowledge(self, request):
00271         """Method used to handle DHCP Acknowledge packet
00272         """
00273         ack = WriteBootProtocolPacket(self.configuration)
00274         ack.parameter_order = request.parameter_request_list
00275         ack.transaction_id = request.transaction_id
00276         # ack.next_server_ip_address =
00277         ack.bootp_flags = request.bootp_flags
00278         ack.relay_agent_ip_address = request.relay_agent_ip_address
00279         mac = request.client_mac_address
00280         ack.client_mac_address = mac
00281         requested_ip_address = request.requested_ip_address
00282         ack.client_ip_address = request.client_ip_address or '0.0.0.0'
00283         ack.your_ip_address = self.server.get_ip_address(request)
00284         ack.dhcp_message_type = 'DHCPACK'
00285         self.configuration.debug('acknowledge:\n {}'.format(str(ack).replace('\n', '\n\t')))
00286         self.server.broadcast(ack)
00287
00288     def received_dhcp_inform(self, inform):
00289         """Method used to handle DHCP Inform packet
00290         """
00291         self.configuration.debug('inform:\n {}'.format(str(inform).replace('\n', '\n\t')))
00292         self.close()
00293         self.server.client_has_chosen(inform)
00294
00295     class DHCPServerConfiguration(object):
00296         """Class to load DHCP server configuration from file or command line
00297         """
00298         dhcp_offer_after_seconds = 10
00299         dhcp_acknowledge_after_seconds = 10
00300         length_of_transaction = 40
00301
00302         network = '192.168.173.0'
00303         broadcast_address = '255.255.255.255'
00304         subnet_mask = '255.255.255.0'
00305         router = None # list of ips
00306         # 1 day is 86400
00307         ip_address_lease_time = 300 # seconds
00308         domain_name_server = None # list of ips
00309
00310         host_file = 'hosts.csv'
00311
00312         debug = lambda *args, **kw: None
00313
00314     def load(self, file):
00315         """Load configuration from file using exec to parse file as object dictionary
00316         or get ALL command line arguments and change them using regexp to file layout
00317         and treat as file
00318         """
00319         if (len(file) > 0 and exists(file)):
00320             with open(file) as f:
00321                 exec(f.read(), self.__dict__)
00322         else:
00323             args = ' '.join(sys.argv[1:])
00324             args = re.sub(' -', '\n', args)
00325             args = re.sub('^-', '\n', args)

```

```

00326         args = re.sub('^[a-z_]+)([ ]+)(.+)$', r"\1=\3", args, flags=re.MULTILINE)
00327         exec(args, self.__dict__)
00328
00329     def adjust_if_this_computer_is_a_router(self):
00330         """Automatically adjust some DHCP configuration parameters if this computer is router
00331         """
00332         ip_addresses = get_host_ip_addresses()
00333         for ip in reversed(ip_addresses):
00334             if ip.split('.')[3] == '1':
00335                 self.router = [ip]
00336                 self.domain_name_server = [ip]
00337                 self.networknetwork = '.'.join(ip.split('.')[:3] + ['0'])
00338                 self.broadcast_addressbroadcast_address = '.'.join(ip.split('.')[:3] + ['255'])
00339                 #self.ip_forwarding_enabled = True
00340                 #self.non_local_source_routing_enabled = True
00341                 #self.perform_mask_discovery = True
00342
00343     def all_ip_addresses(self):
00344         ips = ip_addresses(self.networknetwork, self.subnet_mask)
00345         for i in range(5):
00346             next(ips)
00347         return ips
00348
00349     def network_filter(self):
00350         return NETWORK(self.networknetwork, self.subnet_mask)
00351
00352 def ip_addresses(network, subnet_mask):
00353     import socket, struct
00354     subnet_mask = struct.unpack('>I', socket.inet_aton(subnet_mask))[0]
00355     network = struct.unpack('>I', socket.inet_aton(network))[0]
00356     network = network & subnet_mask
00357     start = network + 1
00358     end = (network | (~subnet_mask & 0xffffffff))
00359     return (socket.inet_ntoa(struct.pack('>I', i)) for i in range(start, end))
00360
00361 class ALL(object):
00362     """Comparator class
00363     """
00364     def __eq__(self, other):
00365         return True
00366     def __repr__(self):
00367         return self.__class__.__name__
00368
00369 ALL = ALL()
00370
00371 class GREATER(object):
00372     """Comparator class
00373     """
00374     def __init__(self, value):
00375         self.value = value
00376     def __eq__(self, other):
00377         return type(self.value)(other) > self.value
00378
00379 class NETWORK(object):
00380     """Comparator class to check if address within same network
00381     """
00382     def __init__(self, network, subnet_mask):
00383         self.subnet_mask = struct.unpack('>I', inet_aton(subnet_mask))[0]
00384         self.network = struct.unpack('>I', inet_aton(network))[0]
00385     def __eq__(self, other):
00386         ip = struct.unpack('>I', inet_aton(other))[0]
00387         return ip & self.subnet_mask == self.network and \
00388             ip - self.network and \
00389             ip - self.network != ~self.subnet_mask & 0xffffffff
00390
00391 class CASEINSENSITIVE(object):
00392     """Comparator class
00393     """
00394     def __init__(self, s):
00395         self.s = s.lower()
00396     def __eq__(self, other):
00397         return self.s == other.lower()
00398
00399 class CSVDatabase(object):
00400     """Class handling CSV file database to keep host definitions
00401     """
00402     delimiter = ','
00403
00404     def __init__(self, file_name):
00405         """Construct new CSV database with storage in file_name
00406         """
00407         self.file_name = file_name
00408         self.file('a').close() # create file
00409
00410     def file(self, mode = 'r'):
00411         """Open CSV file with selected mode
00412         """

```

```

00413         return open(self.file_name, mode)
00414
00415     def get(self, pattern):
00416         """Get CSV entry representing host (MAC) and lease (IP)
00417         """
00418         pattern = list(pattern)
00419         return [line for line in self.all() if pattern == line]
00420
00421     def add(self, line):
00422         """Add host entry to CSV file
00423         """
00424         with self.file('a') as f:
00425             f.write(self.delimiter.join(line) + '\n')
00426
00427     def delete(self, pattern):
00428         """Delete host entry from CSV file
00429         """
00430         lines = self.all()
00431         lines_to_delete = self.get(pattern)
00432         self.file('w').close() # empty file
00433         for line in lines:
00434             if line not in lines_to_delete:
00435                 self.add(line)
00436
00437     def all(self):
00438         """Get all entries from CSV file
00439         """
00440         with self.file() as f:
00441             return [list(line.strip().split(self.delimiter)) for line in f]
00442
00443 class Host(object):
00444     """Class representing host with MAC address, IP, hostname if available and last used timestamp
00445     """
00446     def __init__(self, mac, ip, hostname, last_used):
00447         self.mac = mac.upper()
00448         self.ip = ip
00449         self.hostname = hostname
00450         self.last_used = int(last_used)
00451
00452     @classmethod
00453     def from_tuple(cls, line):
00454         mac, ip, hostname, last_used = line
00455         last_used = int(last_used)
00456         return cls(mac, ip, hostname, last_used)
00457
00458     @classmethod
00459     def from_packet(cls, packet):
00460         return cls(packet.client_mac_address,
00461                    packet.requested_ip_address or packet.client_ip_address,
00462                    packet.host_name or "",
00463                    int(time.time()))
00464
00465     @staticmethod
00466     def get_pattern(mac = ALL, ip = ALL, hostname = ALL, last_used = ALL):
00467         return [mac, ip, hostname, last_used]
00468
00469     def to_tuple(self):
00470         """Convert host to tuple
00471         """
00472         return [self.mac, self.ip, self.hostname, str(int(self.last_used))]
00473
00474     def to_pattern(self):
00475         """Convert host to pattern
00476         """
00477         return self.get_pattern(ip = self.ip, mac = self.mac)
00478
00479     def __hash__(self):
00480         return hash(self.key)
00481
00482     def __eq__(self, other):
00483         return self.to_tuple() == other.to_tuple()
00484
00485     def has_valid_ip(self):
00486         """Check if host has valid IP address
00487         """
00488         return self.ip and self.ip != '0.0.0.0'
00489
00490 class HostDatabase(object):
00491     def __init__(self, file_name):
00492         self.db = CSVDatabase(file_name)
00493
00494     def get(self, **kw):
00495         pattern = Host.get_pattern(**kw)
00496         return list(map(Host.from_tuple, self.db.get(pattern)))
00497
00498     def add(self, host):
00499         self.db.add(host.to_tuple())

```

```

00500
00501     def delete(self, host = None, **kw):
00502         if host is None:
00503             pattern = Host.get_pattern(**kw)
00504         else:
00505             pattern = host.to_pattern()
00506         self.db.delete(pattern)
00507
00508     def all(self):
00509         return list(map(Host.from_tuple, self.db.all()))
00510
00511     def replace(self, host):
00512         self.delete(host)
00513         self.add(host)
00514
00515     def sorted_hosts(hosts):
00516         hosts = list(hosts)
00517         hosts.sort(key = lambda host: (host.hostname.lower(), host.mac.lower(), host.ip.lower()))
00518         return hosts
00519
00520     class DHCPServer(object):
00521
00522         def __init__(self, configuration = None):
00523             if configuration == None:
00524                 configuration = DHCPServerConfiguration()
00525
00526             self.configuration = configuration
00527             self.socket = socket(type = SOCK_DGRAM)
00528             self.socket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
00529             self.socket.bind(('', 67))
00530             self.delay_worker = TransactionDelayWorker()
00531             self.closed = False
00532             self.transactions = collections.defaultdict(lambda: DHCPTransaction(self)) # id: transaction
00533             self.hosts = HostDatabase(self.configuration.host_file)
00534             self.time_started = time.time()
00535
00536         def close(self):
00537             self.socket.close()
00538             self.closed = True
00539             self.delay_worker.close()
00540             for transaction in list(self.transactions.values()):
00541                 transaction.close()
00542
00543         def update(self, timeout = 0):
00544             try:
00545                 reads = select.select([self.socket], [], [], timeout)[0]
00546             except ValueError:
00547                 # ValueError: file descriptor cannot be a negative integer (-1)
00548                 return
00549             for socket in reads:
00550                 try:
00551                     packet = ReadBootProtocolPacket(*socket.recvfrom(4096))
00552                 except OSError:
00553                     # OSError: [WinError 10038] An operation was attempted on something that is not a
socket
00554                     pass
00555                 else:
00556                     self.received(packet)
00557             for transaction_id, transaction in list(self.transactions.items()):
00558                 if transaction.is_done():
00559                     transaction.close()
00560                 self.transactions.pop(transaction_id)
00561
00562         def received(self, packet):
00563             if not self.transactions[packet.transaction_id].receive(packet):
00564                 self.configuration.debug('received:\n {}'.format(str(packet).replace('\n', '\n\t')))
00565
00566         def client_has_chosen(self, packet):
00567             self.configuration.debug('client_has_chosen:\n {}'.format(str(packet).replace('\n', '\n\t')))
00568             host = Host.from_packet(packet)
00569             if not host.has_valid_ip():
00570                 return
00571             self.hosts.replace(host)
00572
00573         def is_valid_client_address(self, address):
00574             if address is None:
00575                 return False
00576             a = address.split('.')
00577             s = self.configuration.subnet_mask.split('.')
00578             n = self.configuration.network.split('.')
00579             return all(s[i] == '0' or a[i] == n[i] for i in range(4))
00580
00581         def get_ip_address(self, packet):
00582             mac_address = packet.client_mac_address
00583             requested_ip_address = packet.requested_ip_address
00584             known_hosts = self.hosts.get(mac = CASEINSENSITIVE(mac_address))
00585             ip = None

```



```

00586         if known_hosts:
00587             # 1. choose known ip address
00588             for host in known_hosts:
00589                 if self.is_valid_client_address(host.ip):
00590                     ip = host.ip
00591             print('known ip:', ip)
00592         if ip is None and self.is_valid_client_address(requested_ip_address):
00593             # 2. choose valid requested ip address
00594             ip = requested_ip_address
00595             print('valid ip:', ip)
00596         if ip is None:
00597             # 3. choose new, free ip address
00598             chosen = False
00599             network_hosts = self.hosts.get(ip = self.configuration.network_filter())
00600             for ip in self.configuration.all_ip_addresses():
00601                 if not any(host.ip == ip for host in network_hosts):
00602                     chosen = True
00603                     break
00604             if not chosen:
00605                 # 4. reuse old valid ip address
00606                 network_hosts.sort(key = lambda host: host.last_used)
00607                 ip = network_hosts[0].ip
00608                 assert self.is_valid_client_address(ip)
00609             print('new ip:', ip)
00610         if not any([host.ip == ip for host in known_hosts]):
00611             print('add', mac_address, ip, packet.host_name)
00612             self.hosts.replace(Host(mac_address, ip, packet.host_name or "", time.time()))
00613         return ip
00614
00615     @property
00616     def server_identifiers(self):
00617         return get_host_ip_addresses()
00618
00619     def broadcast(self, packet):
00620         self.configuration.debug('broadcasting:\n {}'.format(str(packet).replace('\n', '\n\t')))
00621         for addr in self.server_identifiers:
00622             broadcast_socket = socket(type = SOCK_DGRAM)
00623             broadcast_socket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
00624             broadcast_socket.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)
00625             packet.server_identifier = addr
00626             broadcast_socket.bind((addr, 67))
00627             try:
00628                 data = packet.to_bytes()
00629                 broadcast_socket.sendto(data, ('255.255.255.255', 68))
00630                 broadcast_socket.sendto(data, (addr, 68))
00631             finally:
00632                 broadcast_socket.close()
00633
00634     def run(self):
00635         while not self.closed:
00636             try:
00637                 self.update(1)
00638             except KeyboardInterrupt:
00639                 break
00640             except:
00641                 traceback.print_exc()
00642
00643     def run_in_thread(self):
00644         thread = threading.Thread(target = self.run)
00645         thread.start()
00646         return thread
00647
00648     def debug_clients(self):
00649         for line in self.ips.all():
00650             line = '\t'.join(line)
00651             if line:
00652                 self.configuration.debug(line)
00653
00654     def get_all_hosts(self):
00655         return sorted_hosts(self.hosts.get())
00656
00657     def get_current_hosts(self):
00658         return sorted_hosts(self.hosts.get(last_used = GREATER(self.time_started)))
00659
00660 class ThreadedTcpRequestHandler(socketserver.StreamRequestHandler):
00661     """Control socket client connection handler
00662     """
00663     def handle(self):
00664         """Method used to handle client connection parsing commands and giving response to them
00665         """
00666         self.request.sendall(bytes("Welcome to micro python dhcp server", 'ascii'))
00667         try:
00668             while(True):
00669                 self.request.sendall(bytes("\r\npydhcp ?> ", 'ascii'))
00670                 data = self.rfile.readline().strip()
00671                 if(data.decode() == "hosts"):
00672                     self.request.sendall(bytes("Active

```



```

00036     s = base64.b16encode(data)
00037     return ':'.join([s[i:i+2].decode('ascii') for i in range(0, 12, 2)])
00038
00039 def macpack(mac):
00040     return base64.b16decode(mac.replace(':', '').replace('-', '').encode('ascii'))
00041
00042 def unpackbool(data):
00043     return data[0]
00044
00045 def packbool(bool):
00046     return bytes([bool])
00047
00048 options = [
00049     # RFC1497 vendor extensions
00050     ('pad', None, None),
00051     ('subnet_mask', inet_ntoa, inet_aton),
00052     ('time_offset', None, None),
00053     ('router', inet_ntoaX, inet_atonX),
00054     ('time_server', inet_ntoaX, inet_atonX),
00055     ('name_server', inet_ntoaX, inet_atonX),
00056     ('domain_name_server', inet_ntoaX, inet_atonX),
00057     ('log_server', inet_ntoaX, inet_atonX),
00058     ('cookie_server', inet_ntoaX, inet_atonX),
00059     ('lpr_server', inet_ntoaX, inet_atonX),
00060     ('impress_server', inet_ntoaX, inet_atonX),
00061     ('resource_location_server', inet_ntoaX, inet_atonX),
00062     ('host_name', lambda d: d.decode('ASCII'), lambda d: d.encode('ASCII')),
00063     ('boot_file_size', None, None),
00064     ('merit_dump_file', None, None),
00065     ('domain_name', None, None),
00066     ('swap_server', inet_ntoa, inet_aton),
00067     ('root_path', None, None),
00068     ('extensions_path', None, None),
00069     # IP Layer Parameters per Host
00070     ('ip_forwarding_enabled', unpackbool, packbool),
00071     ('non_local_source_routing_enabled', unpackbool, packbool),
00072     ('policy_filer', None, None),
00073     ('maximum_datagram_reassembly_size', shortunpack, shortpack),
00074     ('default_ip_time_to_live', lambda data: data[0], lambda i: bytes([i])),
00075     ('path_mtu_aging_timeout', None, None),
00076     ('path_mtu_plateau_table', None, None),
00077     # IP Layer Parameters per Interface
00078     ('interface_mtu', None, None),
00079     ('all_subnets_are_local', unpackbool, packbool),
00080     ('broadcast_address', inet_ntoa, inet_aton),
00081     ('perform_mask_discovery', unpackbool, packbool),
00082     ('mask_supplier', None, None),
00083     ('perform_router_discovery', None, None),
00084     ('router_solicitation_address', inet_ntoa, inet_aton),
00085     ('static_route', None, None),
00086     # Link Layer Parameters per Interface
00087     ('trailer_encapsulation_option', None, None),
00088     ('arp_cache_timeout', None, None),
00089     ('ethernet_encapsulation', None, None),
00090     # TCP Parameters
00091     ('tcp_default_ttl', None, None),
00092     ('tcp_keep_alive_interval', None, None),
00093     ('tcp_keep_alive_garbage', None, None),
00094     # Application and Service Parameters Part 1
00095     ('network_information_service_domain', None, None),
00096     ('network_informtaion_servers', inet_ntoaX, inet_atonX),
00097     ('network_time_protocol_servers', inet_ntoaX, inet_atonX),
00098     ('vendor_specific_information', None, None),
00099     ('netbios_over_tcp_ip_name_server', inet_ntoaX, inet_atonX),
00100     ('netbios_over_tcp_ip_datagram_distribution_server', inet_ntoaX, inet_atonX),
00101     ('netbios_over_tcp_ip_node_type', None, None),
00102     ('netbios_over_tcp_ip_scope', None, None),
00103     ('x_window_system_font_server', inet_ntoaX, inet_atonX),
00104     ('x_window_system_display_manager', inet_ntoaX, inet_atonX),
00105     # DHCP Extensions
00106     ('requested_ip_address', inet_ntoa, inet_aton),
00107     ('ip_address_lease_time', lambda d: struct.unpack('>I', d)[0], lambda i: struct.pack('>I', i)),
00108     ('option_overload', None, None),
00109     ('dhcp_message_type', lambda data: dhcp_message_types.get(data[0], data[0]), (lambda name:
bytes([reversed_dhcp_message_types.get(name, name)]))),
00110     ('server_identifier', inet_ntoa, inet_aton),
00111     ('parameter_request_list', list, bytes),
00112     ('message', None, None),
00113     ('maximum_dhcp_message_size', shortunpack, shortpack),
00114     ('renewal_time_value', None, None),
00115     ('rebinding_time_value', None, None),
00116     ('vendor_class_identifier', None, None),
00117     ('client_identifier', macunpack, macpack),
00118     ('tftp_server_name', None, None),
00119     ('boot_file_name', None, None),
00120     # Application and Service Parameters Part 2
00121     ('network_information_service_domain', None, None),

```

```

00122     ('network_information_servers', inet_ntoaX, inet_atonX),
00123     ('', None, None),
00124     ('', None, None),
00125     ('mobile_ip_home_agent', inet_ntoaX, inet_atonX),
00126     ('smtp_server', inet_ntoaX, inet_atonX),
00127     ('pop_servers', inet_ntoaX, inet_atonX),
00128     ('nntp_server', inet_ntoaX, inet_atonX),
00129     ('default_www_server', inet_ntoaX, inet_atonX),
00130     ('default_finger_server', inet_ntoaX, inet_atonX),
00131     ('default_irc_server', inet_ntoaX, inet_atonX),
00132     ('streettalk_server', inet_ntoaX, inet_atonX),
00133     ('stda_server', inet_ntoaX, inet_atonX),
00134     ]
00135
00136 assert options[18][0] == 'extensions_path', options[18][0]
00137 assert options[25][0] == 'path_mtu_plateau_table', options[25][0]
00138 assert options[33][0] == 'static_route', options[33][0]
00139 assert options[50][0] == 'requested_ip_address', options[50][0]
00140 assert options[64][0] == 'network_information_service_domain', options[64][0]
00141 assert options[76][0] == 'stda_server', options[76][0]
00142
00143
00144 class ReadBootProtocolPacket(object):
00145
00146     for i, o in enumerate(options):
00147         locals()[o[0]] = None
00148         locals()['option_{0}'.format(i)] = None
00149
00150     del i, o
00151
00152     def __init__(self, data, address = ('0.0.0.0', 0)):
00153         self.data = data
00154         self.address = address
00155         self.host = address[0]
00156         self.port = address[1]
00157
00158         # wireshark = wikipedia = data[...]
00159
00160         self.message_type = self.OP = data[0]
00161         self.hardware_type = self.HTYPE = data[1]
00162         self.hardware_address_length = self.HLEN = data[2]
00163         self.hops = self.HOPS = data[3]
00164
00165         self.XID = self.transaction_id = struct.unpack('>I', data[4:8])[0]
00166
00167         self.seconds_elapsed = self.SECONDS = shortunpack(data[8:10])
00168         self.bootp_flags = self.FLAGS = shortunpack(data[8:10])
00169
00170         self.client_ip_address = self.CIADDR = inet_ntoa(data[12:16])
00171         self.your_ip_address = self.YIADDR = inet_ntoa(data[16:20])
00172         self.next_server_ip_address = self.SIADDR = inet_ntoa(data[20:24])
00173         self.relay_agent_ip_address = self.GIADDR = inet_ntoa(data[24:28])
00174
00175         self.client_mac_address = self.CHADDR = macunpack(data[28: 28 + self.hardware_address_length])
00176         index = 236
00177         self.magic_cookie = self.magic_cookie = inet_ntoa(data[index:index + 4]); index += 4
00178         self.options = dict()
00179         self.named_options = dict()
00180         while index < len(data):
00181             option = data[index]; index += 1
00182             if option == 0:
00183                 # padding
00184                 # Can be used to pad other options so that they are aligned to the word boundary; is
not followed by length byte
00185                 continue
00186             if option == 255:
00187                 # end
00188                 break
00189             option_length = data[index]; index += 1
00190             option_data = data[index: index + option_length]; index += option_length
00191             self.options[option] = option_data
00192             if option < len(options):
00193                 option_name, function, _ = options[option]
00194                 if function:
00195                     option_data = function(option_data)
00196                 if option_name:
00197                     setattr(self, option_name, option_data)
00198                     self.named_options[option_name] = option_data
00199                 setattr(self, 'option_{0}'.format(option), option_data)
00200
00201     def __getitem__(self, key):
00202         print(key, dir(self))
00203         return getattr(self, key, None)
00204
00205     def __contains__(self, key):
00206         return key in self.__dict__
00207

```


8.7 ttlDict.py

[Go to the documentation of this file.](#)

```

00001 from collections import OrderedDict
00002 from threading import RLock
00003 import time
00004
00005 __all__ = ['TTLOrderedDict']
00006
00007
00008 class TTLOrderedDict(OrderedDict):
00009     """
00010     OrderedDict with TTL
00011     Extra args and kwargs are passed to initial .update() call
00012     """
00013     def __init__(self, default_ttl, *args, **kwargs):
00014         """
00015         Be warned, if you use this with Python versions earlier than 3.6
00016         when passing **kwargs order is not preseverd.
00017         """
00018         assert isinstance(default_ttl, int)
00019         self._default_ttl = default_ttl
00020         self._lock = RLock()
00021         super().__init__()
00022         self.update(*args, **kwargs)
00023
00024     def __repr__(self):
00025         return '<TTLOrderedDict@%#08x; ttl=%r, OrderedDict=%r;>' % (
00026             id(self), self._default_ttl, self.items())
00027
00028     def __len__(self):
00029         with self._lock:
00030             self._purge()
00031             return super().__len__()
00032
00033     def set_ttl(self, key, ttl, now=None):
00034         """Set TTL for the given key"""
00035         if now is None:
00036             now = time.time()
00037         with self._lock:
00038             value = self[key]
00039             super().__setitem__(key, (now + ttl, value))
00040
00041     def get_ttl(self, key, now=None):
00042         """Return remaining TTL for a key"""
00043         if now is None:
00044             now = time.time()
00045         with self._lock:
00046             expire, _value = super().__getitem__(key)
00047             return expire - now
00048
00049     def expire_at(self, key, timestamp):
00050         """Set the key expire timestamp"""
00051         with self._lock:
00052             value = self.__getitem__(key)
00053             super().__setitem__(key, (timestamp, value))
00054
00055     def is_expired(self, key, now=None):
00056         """ Check if key has expired, and return it if so"""
00057         with self._lock:
00058             if now is None:
00059                 now = time.time()
00060
00061             expire, _value = super().__getitem__(key)
00062
00063             if expire:
00064                 if expire < now:
00065                     return key
00066
00067     def _purge(self):
00068         _keys = list(super().__iter__())
00069         _remove = [key for key in _keys if self.is_expired(key)] # noqa
00070         [self.__delitem__(key) for key in _remove]
00071
00072     def __iter__(self):
00073         """
00074         Yield only non expired keys, without purging the expired ones
00075         """
00076         with self._lock:
00077             for key in super().__iter__():
00078                 if not self.is_expired(key):
00079                     yield key
00080
00081     def __setitem__(self, key, value):
00082         with self._lock:

```

```
00083         if self._default_ttl is None:
00084             expire = None
00085         else:
00086             expire = time.time() + self._default_ttl
00087         super().__setitem__(key, (expire, value))
00088
00089     def __delitem__(self, key):
00090         with self._lock:
00091             super().__delitem__(key)
00092
00093     def __getitem__(self, key):
00094         with self._lock:
00095             if self.is_expired(key):
00096                 self.__delitem__(key)
00097                 raise KeyError
00098             item = super().__getitem__(key)[1]
00099             return item
00100
00101     def keys(self):
00102         with self._lock:
00103             self._purge()
00104             return super().keys()
00105
00106     def items(self):
00107         with self._lock:
00108             self._purge()
00109             _items = list(super(OrderedDict, self).items())
00110             return [(k, v[1]) for (k, v) in _items]
00111
00112     def values(self):
00113         with self._lock:
00114             self._purge()
00115             _values = list(super(OrderedDict, self).values())
00116             return [v[1] for v in _values]
00117
00118     def get(self, key, default=None):
00119         try:
00120             return self[key]
00121         except KeyError:
00122             return default
```


Index

- `__contains__`
 - `listener.ReadBootProtocolPacket`, 50
 - `__delitem__`
 - `ttlDict.TTLOrderedDict`, 62
 - `__eq__`
 - `dhcp.ALL`, 19
 - `dhcp.CASEINSENSITIVE`, 21
 - `dhcp.GREATER`, 39
 - `dhcp.Host`, 40
 - `dhcp.NETWORK`, 46
 - `__getitem__`
 - `listener.ReadBootProtocolPacket`, 50
 - `ttlDict.TTLOrderedDict`, 62
 - `__gt__`
 - `listener.ReadBootProtocolPacket`, 50
 - `__hash__`
 - `dhcp.Host`, 41
 - `__init__`
 - `dhcp.CASEINSENSITIVE`, 20
 - `dhcp.CSVDatabase`, 22
 - `dhcp.DHCPServer`, 25
 - `dhcp.DHCPTransaction`, 34
 - `dhcp.GREATER`, 38
 - `dhcp.Host`, 40
 - `dhcp.HostDatabase`, 44
 - `dhcp.NETWORK`, 46
 - `dhcp.PriorityQueue`, 47
 - `dhcp.TransactionDelayWorker`, 60
 - `dhcp.WriteBootProtocolPacket`, 66
 - `listener.ReadBootProtocolPacket`, 49
 - `ttlDict.TTLOrderedDict`, 62
 - `__iter__`
 - `ttlDict.TTLOrderedDict`, 62
 - `__len__`
 - `ttlDict.TTLOrderedDict`, 63
 - `__repr__`
 - `dhcp.ALL`, 19
 - `ttlDict.TTLOrderedDict`, 63
 - `__setitem__`
 - `ttlDict.TTLOrderedDict`, 63
 - `__str__`
 - `dhcp.WriteBootProtocolPacket`, 67
 - `listener.ReadBootProtocolPacket`, 50
- `acknowledge`
 - `dhcp.DHCPTransaction`, 34
- `add`
 - `dhcp.CSVDatabase`, 22
 - `dhcp.HostDatabase`, 44
- `address`
 - `listener.ReadBootProtocolPacket`, 51
- `adjust_if_this_computer_is_a_router`
 - `dhcp.DHCPServerConfiguration`, 30
- `ALL`
 - `dhcp`, 12
- `all`
 - `dhcp.CSVDatabase`, 22
 - `dhcp.HostDatabase`, 44
- `all_ip_addresses`
 - `dhcp.DHCPServerConfiguration`, 30
- `bootp_flags`
 - `dhcp.WriteBootProtocolPacket`, 68
 - `listener.ReadBootProtocolPacket`, 51
- `broadcast`
 - `dhcp.DHCPServer`, 25
- `broadcast_address`
 - `dhcp.DHCPServerConfiguration`, 31
- `CHADDR`
 - `listener.ReadBootProtocolPacket`, 51
- `CIADDR`
 - `listener.ReadBootProtocolPacket`, 51
- `client_has_chosen`
 - `dhcp.DHCPServer`, 25
- `client_ip_address`
 - `dhcp.WriteBootProtocolPacket`, 68
 - `listener.ReadBootProtocolPacket`, 51
- `client_mac_address`
 - `dhcp.WriteBootProtocolPacket`, 68
 - `listener.ReadBootProtocolPacket`, 51
- `close`
 - `dhcp.DHCPServer`, 25
 - `dhcp.DHCPTransaction`, 35
 - `dhcp.TransactionDelayWorker`, 60
- `closed`
 - `dhcp.DHCPServer`, 28
 - `dhcp.TransactionDelayWorker`, 60
- `configuration`
 - `dhcp`, 13
 - `dhcp.DHCPServer`, 28
 - `dhcp.DHCPTransaction`, 37
 - `dhcp.ThreadedTcpServer`, 58
- `data`
 - `listener`, 15
 - `listener.ReadBootProtocolPacket`, 52
- `db`
 - `dhcp.HostDatabase`, 45
- `debug`

- dhcp, 13
 - dhcp.DHCPServerConfiguration, 31
 - dhcp.DHCPTransaction, 37
- debug_clients
 - dhcp.DHCPServer, 26
- debug_msg
 - dhcp, 12
- delay_worker
 - dhcp.DHCPServer, 28
- delete
 - dhcp.CSVDatabase, 23
 - dhcp.HostDatabase, 44
- delimiter
 - dhcp.CSVDatabase, 24
- dhcp, 11
 - ALL, 12
 - configuration, 13
 - debug, 13
 - debug_msg, 12
 - get_host_ip_addresses, 12
 - ip_address_lease_time, 13
 - ip_addresses, 12
 - messages, 13
 - server, 13
 - sorted_hosts, 12
- dhcp.ALL, 19
 - __eq__, 19
 - __repr__, 19
- dhcp.CASEINSENSITIVE, 20
 - __eq__, 21
 - __init__, 20
 - s, 21
- dhcp.CSVDatabase, 21
 - __init__, 22
 - add, 22
 - all, 22
 - delete, 23
 - delimiter, 24
 - file, 23
 - file_name, 24
 - get, 23
- dhcp.DHCPServer, 24
 - __init__, 25
 - broadcast, 25
 - client_has_chosen, 25
 - close, 25
 - closed, 28
 - configuration, 28
 - debug_clients, 26
 - delay_worker, 28
 - get_all_hosts, 26
 - get_current_hosts, 26
 - get_ip_address, 26
 - hosts, 28
 - is_valid_client_address, 26
 - received, 27
 - run, 27
 - run_in_thread, 27
 - server_identifiers, 27
 - socket, 28
 - time_started, 29
 - transactions, 29
 - update, 27
- dhcp.DHCPServerConfiguration, 29
 - adjust_if_this_computer_is_a_router, 30
 - all_ip_addresses, 30
 - broadcast_address, 31
 - debug, 31
 - dhcp_acknowledge_after_seconds, 31
 - dhcp_offer_after_seconds, 32
 - domain_name_server, 32
 - host_file, 32
 - ip_address_lease_time, 32
 - length_of_transaction, 32
 - load, 30
 - network, 32, 33
 - network_filter, 31
 - router, 33
 - subnet_mask, 33
- dhcp.DHCPTransaction, 33
 - __init__, 34
 - acknowledge, 34
 - close, 35
 - configuration, 37
 - debug, 37
 - do_after, 37
 - done, 37
 - done_time, 37
 - is_done, 35
 - packets, 37
 - receive, 35
 - received_dhcp_discover, 35
 - received_dhcp_inform, 36
 - received_dhcp_request, 36
 - send_offer, 36
 - server, 38
- dhcp.GREATER, 38
 - __eq__, 39
 - __init__, 38
 - value, 39
- dhcp.Host, 39
 - __eq__, 40
 - __hash__, 41
 - __init__, 40
 - from_packet, 41
 - from_tuple, 41
 - get_pattern, 41
 - has_valid_ip, 41
 - hostname, 42
 - ip, 42
 - last_used, 43
 - mac, 43
 - to_pattern, 42
 - to_tuple, 42
- dhcp.HostDatabase, 43
 - __init__, 44

- add, [44](#)
- all, [44](#)
- db, [45](#)
- delete, [44](#)
- get, [44](#)
- replace, [45](#)
- dhcp.NETWORK, [45](#)
 - __eq__, [46](#)
 - __init__, [46](#)
 - network, [46](#)
 - subnet_mask, [46](#)
- dhcp.PriorityQueue, [47](#)
 - __init__, [47](#)
 - get, [48](#)
 - put, [48](#)
 - qsize, [48](#)
- dhcp.py, [71](#)
- dhcp.ThreadedTcpRequestHandler, [56](#)
 - handle, [56](#)
- dhcp.ThreadedTcpServer, [57](#)
 - configuration, [58](#)
 - events, [58](#)
 - hosts, [59](#)
 - setConfiguration, [57](#)
 - setEvents, [58](#)
 - setHosts, [58](#)
- dhcp.TransactionDelayWorker, [59](#)
 - __init__, [60](#)
 - close, [60](#)
 - closed, [60](#)
 - do_after, [60](#)
 - queue, [61](#)
 - thread, [61](#)
- dhcp.WriteBootProtocolPacket, [65](#)
 - __init__, [66](#)
 - __str__, [67](#)
 - bootp_flags, [68](#)
 - client_ip_address, [68](#)
 - client_mac_address, [68](#)
 - get_option, [67](#)
 - hardware_address_length, [68](#)
 - hardware_type, [68](#)
 - hops, [68](#)
 - magic_cookie, [69](#)
 - message_type, [69](#)
 - next_server_ip_address, [69](#)
 - options, [67](#)
 - parameter_order, [69](#)
 - relay_agent_ip_address, [69](#)
 - seconds_elapsed, [69](#)
 - to_bytes, [67](#)
 - transaction_id, [70](#)
 - your_ip_address, [70](#)
- dhcp_acknowledge_after_seconds
 - dhcp.DHCPServerConfiguration, [31](#)
- dhcp_message_types
 - listener, [15](#)
- dhcp_offer_after_seconds
 - dhcp.DHCPServerConfiguration, [32](#)
- do_after
 - dhcp.DHCPTransaction, [37](#)
 - dhcp.TransactionDelayWorker, [60](#)
- domain_name_server
 - dhcp.DHCPServerConfiguration, [32](#)
- done
 - dhcp.DHCPTransaction, [37](#)
- done_time
 - dhcp.DHCPTransaction, [37](#)
- events
 - dhcp.ThreadedTcpServer, [58](#)
- expire_at
 - ttldict.TTLOrderedDict, [63](#)
- file
 - dhcp.CSVDatabase, [23](#)
- file_name
 - dhcp.CSVDatabase, [24](#)
- FLAGS
 - listener.ReadBootProtocolPacket, [52](#)
- formatted_named_options
 - listener.ReadBootProtocolPacket, [50](#)
- from_packet
 - dhcp.Host, [41](#)
- from_tuple
 - dhcp.Host, [41](#)
- get
 - dhcp.CSVDatabase, [23](#)
 - dhcp.HostDatabase, [44](#)
 - dhcp.PriorityQueue, [48](#)
 - ttldict.TTLOrderedDict, [63](#)
- get_all_hosts
 - dhcp.DHCPServer, [26](#)
- get_current_hosts
 - dhcp.DHCPServer, [26](#)
- get_host_ip_addresses
 - dhcp, [12](#)
- get_ip_address
 - dhcp.DHCPServer, [26](#)
- get_option
 - dhcp.WriteBootProtocolPacket, [67](#)
- get_pattern
 - dhcp.Host, [41](#)
- get_ttl
 - ttldict.TTLOrderedDict, [64](#)
- GIADDR
 - listener.ReadBootProtocolPacket, [52](#)
- handle
 - dhcp.ThreadedTcpRequestHandler, [56](#)
- hardware_address_length
 - dhcp.WriteBootProtocolPacket, [68](#)
 - listener.ReadBootProtocolPacket, [52](#)
- hardware_type
 - dhcp.WriteBootProtocolPacket, [68](#)
 - listener.ReadBootProtocolPacket, [52](#)

- has_valid_ip
 - dhcp.Host, 41
- HLEN
 - listener.ReadBootProtocolPacket, 52
- HOPS
 - listener.ReadBootProtocolPacket, 53
- hops
 - dhcp.WriteBootProtocolPacket, 68
 - listener.ReadBootProtocolPacket, 53
- host
 - listener.ReadBootProtocolPacket, 53
- host_file
 - dhcp.DHCPServerConfiguration, 32
- hostname
 - dhcp.Host, 42
- hosts
 - dhcp.DHCPServer, 28
 - dhcp.ThreadedTcpServer, 59
- HTYPE
 - listener.ReadBootProtocolPacket, 53
- inet_atonX
 - listener, 14
- inet_ntoaX
 - listener, 14
- ip
 - dhcp.Host, 42
- ip_address_lease_time
 - dhcp, 13
 - dhcp.DHCPServerConfiguration, 32
- ip_addresses
 - dhcp, 12
- is_done
 - dhcp.DHCPTransaction, 35
- is_expired
 - ttlDict.TTLOrderedDict, 64
- is_valid_client_address
 - dhcp.DHCPServer, 26
- items
 - ttlDict.TTLOrderedDict, 64
- keys
 - ttlDict.TTLOrderedDict, 64
- last_used
 - dhcp.Host, 43
- length_of_transaction
 - dhcp.DHCPServerConfiguration, 32
- listener, 14
 - data, 15
 - dhcp_message_types, 15
 - inet_atonX, 14
 - inet_ntoaX, 14
 - macpack, 14
 - macunpack, 15
 - options, 16
 - p, 16
 - packbool, 15
 - packet, 16
 - reads, 16
 - reversed_dhcp_message_types, 16
 - s1, 17
 - shortpack, 17
 - shortunpack, 17
 - unpackbool, 15
- listener.py, 80
- listener.ReadBootProtocolPacket, 48
 - __contains__, 50
 - __getitem__, 50
 - __gt__, 50
 - __init__, 49
 - __str__, 50
 - address, 51
 - bootp_flags, 51
 - CHADDR, 51
 - CIADDR, 51
 - client_ip_address, 51
 - client_mac_address, 51
 - data, 52
 - FLAGS, 52
 - formatted_named_options, 50
 - GIADDR, 52
 - hardware_address_length, 52
 - hardware_type, 52
 - HLEN, 52
 - HOPS, 53
 - hops, 53
 - host, 53
 - HTYPE, 53
 - magic_cookie, 53
 - message_type, 53
 - named_options, 54
 - next_server_ip_address, 54
 - OP, 54
 - options, 54
 - port, 54
 - relay_agent_ip_address, 54
 - seconds_elapsed, 55
 - SECS, 55
 - SIADDR, 55
 - transaction_id, 55
 - XID, 55
 - YIADDR, 55
 - your_ip_address, 56
- load
 - dhcp.DHCPServerConfiguration, 30
- mac
 - dhcp.Host, 43
- macpack
 - listener, 14
- macunpack
 - listener, 15
- magic_cookie
 - dhcp.WriteBootProtocolPacket, 69
 - listener.ReadBootProtocolPacket, 53
- message_type
 - dhcp.WriteBootProtocolPacket, 69

- listener.ReadBootProtocolPacket, 53
- messages
 - dhcp, 13
- named_options
 - listener.ReadBootProtocolPacket, 54
- network
 - dhcp.DHCPServerConfiguration, 32, 33
 - dhcp.NETWORK, 46
- network_filter
 - dhcp.DHCPServerConfiguration, 31
- next_server_ip_address
 - dhcp.WriteBootProtocolPacket, 69
 - listener.ReadBootProtocolPacket, 54
- OP
 - listener.ReadBootProtocolPacket, 54
- options
 - dhcp.WriteBootProtocolPacket, 67
 - listener, 16
 - listener.ReadBootProtocolPacket, 54
- p
 - listener, 16
- packbool
 - listener, 15
- packet
 - listener, 16
- packets
 - dhcp.DHCPTransaction, 37
- parameter_order
 - dhcp.WriteBootProtocolPacket, 69
- port
 - listener.ReadBootProtocolPacket, 54
- put
 - dhcp.PriorityQueue, 48
- qsize
 - dhcp.PriorityQueue, 48
- queue
 - dhcp.TransactionDelayWorker, 61
- README.md, 84
- reads
 - listener, 16
- receive
 - dhcp.DHCPTransaction, 35
- received
 - dhcp.DHCPServer, 27
- received_dhcp_discover
 - dhcp.DHCPTransaction, 35
- received_dhcp_inform
 - dhcp.DHCPTransaction, 36
- received_dhcp_request
 - dhcp.DHCPTransaction, 36
- relay_agent_ip_address
 - dhcp.WriteBootProtocolPacket, 69
 - listener.ReadBootProtocolPacket, 54
- replace
 - dhcp.HostDatabase, 45
 - reversed_dhcp_message_types
 - listener, 16
 - router
 - dhcp.DHCPServerConfiguration, 33
 - run
 - dhcp.DHCPServer, 27
 - run_in_thread
 - dhcp.DHCPServer, 27
- s
 - dhcp.CASEINSENSITIVE, 21
- s1
 - listener, 17
- seconds_elapsed
 - dhcp.WriteBootProtocolPacket, 69
 - listener.ReadBootProtocolPacket, 55
- SECS
 - listener.ReadBootProtocolPacket, 55
- send_offer
 - dhcp.DHCPTransaction, 36
- server
 - dhcp, 13
 - dhcp.DHCPTransaction, 38
- server_identifiers
 - dhcp.DHCPServer, 27
- set_ttl
 - ttlDict.TTLOrderedDict, 65
- setConfiguration
 - dhcp.ThreadedTcpServer, 57
- setEvents
 - dhcp.ThreadedTcpServer, 58
- setHosts
 - dhcp.ThreadedTcpServer, 58
- shortpack
 - listener, 17
- shortunpack
 - listener, 17
- SIADDR
 - listener.ReadBootProtocolPacket, 55
- socket
 - dhcp.DHCPServer, 28
- sorted_hosts
 - dhcp, 12
- subnet_mask
 - dhcp.DHCPServerConfiguration, 33
 - dhcp.NETWORK, 46
- thread
 - dhcp.TransactionDelayWorker, 61
- time_started
 - dhcp.DHCPServer, 29
- to_bytes
 - dhcp.WriteBootProtocolPacket, 67
- to_pattern
 - dhcp.Host, 42
- to_tuple
 - dhcp.Host, 42
- transaction_id

- dhcp.WriteBootProtocolPacket, [70](#)
 - listener.ReadBootProtocolPacket, [55](#)
- transactions
 - dhcp.DHCPsServer, [29](#)
- ttldict, [17](#)
- ttldict.py, [84](#)
- ttldict.TTLOrderedDict, [61](#)
 - __delitem__, [62](#)
 - __getitem__, [62](#)
 - __init__, [62](#)
 - __iter__, [62](#)
 - __len__, [63](#)
 - __repr__, [63](#)
 - __setitem__, [63](#)
 - expire_at, [63](#)
 - get, [63](#)
 - get_ttl, [64](#)
 - is_expired, [64](#)
 - items, [64](#)
 - keys, [64](#)
 - set_ttl, [65](#)
 - values, [65](#)
- unpackbool
 - listener, [15](#)
- update
 - dhcp.DHCPsServer, [27](#)
- value
 - dhcp.GREATER, [39](#)
- values
 - ttldict.TTLOrderedDict, [65](#)
- XID
 - listener.ReadBootProtocolPacket, [55](#)
- YIADDR
 - listener.ReadBootProtocolPacket, [55](#)
- your_ip_address
 - dhcp.WriteBootProtocolPacket, [70](#)
 - listener.ReadBootProtocolPacket, [56](#)