

---

# **async-dnsproxy**

*Release 0.1.1*

**Jan 15, 2020**



---

## Contents

---

<b>1</b>	<b>Installation &amp; Testing</b>	<b>1</b>
<b>2</b>	<b>Modules Documentation</b>	<b>3</b>
<b>3</b>	<b>Introduction</b>	<b>11</b>
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



### 1.1 Installation

#### From PyPI

```
$ pip install asyncdns
```

#### From source

Download the latest *asyncdns* library from: <https://github.com/abdullahselek/async-dns>

Extract the source distribution and run:

```
$ python setup.py build
$ python setup.py install
```

### 1.2 Running Tests

The test suite can be run against a single Python version which requires `pip install pytest` and optionally `pip install pytest-cov` (these are included if you have installed dependencies from `requirements.testing.txt`)

To run the unit tests with a single Python version:

```
$ py.test tests -s -v
```

to also run code coverage:

```
$ py.test tests -s -v --cov-report html --cov=asyncdns
```

## 1.3 Getting the code

The code is hosted at [Github](#).

Check out the latest development version anonymously with:

```
$ git clone https://github.com/abdullahselek/async-dnsproxy.git
$ cd async-dnsproxy
```

## 2.1 DNS Message

**class** `asyncdns.py.dns_raw_message.DNSRawMessage`

Bases: `object`

Class that creates DNS raw message as byte array.

**query** (*domain*: `str`, *record\_type*: `asyncdns.py.dns_enum.RecordType`, *socket\_type*: `asyncdns.py.dns_enum.SocketType = <SocketType.udp: 2>`)  
Creates a raw dns message query.

```
>>> from asyncdns.py.dns_raw_message import DNSRawMessage
>>> dns_raw_message = DNSRawMessage()
>>> data = dns_raw_message.query('google.com', RecordType.a)
```

### Parameters

- **domain** (`str`) – Domain to query it's DNS details.
- **record\_type** (`RecordType`) – DNS record type.
- **socket\_type** (`SocketType`) – Socket type.

**Return type** bytes.

**class** `asyncdns.py.dns_raw_message.Flag`

Bases: `enum.Enum`

Enum for DNS flags.

**aa** = 1

**ad** = 2

**cd** = 3

**ra** = 4

`rd = 5`

`tc = 6`

**class** `asyncdnsproxy.dns_raw_message.State`

Bases: `enum.Enum`

Enum for raw message states.

`have_answers = 6`

`have_flags = 3`

`have_header = 4`

`have_messageid = 2`

`have_questions = 5`

`initial = 1`

**class** `asyncdnsproxy.dns_message_header.Header`

Bases: `object`

DNS message header class that has header properties.

**additional\_count**

Additional count property of dns message header.

**answer\_count**

Answer count property of dns message header.

**authority\_count**

Authority count property of dns message header.

**flags**

Flags property of dns message header.

**message\_id**

Message id property of dns message header.

**question\_count**

Question count property of dns message header.

**class** `asyncdnsproxy.dns_message_question.Question`

Bases: `object`

DNS message question class that has question details.

**class\_type**

Class type property of dns message question.

**name**

Name property of dns message question.

**question\_type**

Question type property of dns message question.

**class** `asyncdnsproxy.dns_message_resourcerecord.ResourceRecord`

Bases: `object`

Class that holds resource record details.

**class\_type**

Class type property of dns message question.

**get\_ipv4\_address** ()  
Returns ipv4 address if record type A.

**get\_ipv6\_address** ()  
Returns ipv6 address if record type AAAA.

**get\_txt** ()  
Returns readable TXT if record type TXT.

**name**  
Name property of dns message question.

**record\_data**  
Record data property of dns message question.

**record\_data\_length**  
Length of record data property of dns message question.

**record\_type**  
Record type property of dns message question.

**ttl**  
Time to live property of dns message question.

**class** `asyncdns.py.dns_message.DNSMessage`  
Bases: `object`

**additional**  
Additional property of dns message.

**answers**  
Answers property of dns message.

**authority**  
Authority property of dns message.

**header**  
Header property of dns message.

**questions**  
Questions property of dns message.

## 2.2 DNS Message Decoder

**class** `asyncdns.py.dns_message_decoder.DNSMessageDecoder`  
Bases: `object`

Decoder class that decodes raw bytes to meaningful dns message.

**static decode** (*buffer*: `List[T]`, *received\_data*: `List[T]`, *record\_type*:  
`asyncdns.py.dns_enum.RecordType`, *socket*: `asyncdns.py.dns_enum.SocketType =`  
`<SocketType.udp: 2>`)

Decodes dns message response into a readable dns message.

### Parameters

- **buffer** (*list*) – Raw byte list of dns message query.
- **received\_data** (*list*) – Raw byte list of dns message response.
- **record\_type** (*RecordType*) – DNS message record type.
- **socket\_type** (*SocketType*) – DNS client socket type.

**Return type** *DNSMessage*

**static decode\_name** (*bytes: List[T], name\_start\_index: int, name\_length: int*)  
 Decodes list of unsigned 8-bit array to domain str.

**Parameters**

- **bytes** (*list*) – Raw byte list of dns header.
- **name\_start\_index** (*int*) – Starting index of name.
- **name\_length** (*int*) – Length of name.

**Return type** array

**offset** = 0

**static read\_header** (*bytes: List[T]*)  
 Reads byte list and creates a high level dns message header.

**Parameters** **bytes** (*list*) – Raw byte list of dns header.

**Return type** *Header*

**static read\_questions** (*bytes: List[T], count: int*)  
 Reads byte list and creates a high level dns message questions list.

**Parameters**

- **bytes** (*list*) – Raw byte list of dns header.
- **count** (*int*) – Question count.

**Return type** list (list of question/s)

**static read\_resource\_records** (*buffer: List[T], received\_data: List[T], count: int*)  
 Reads byte list and create resource records.

**Parameters**

- **buffer** – Raw byte list of sent dns message.
- **received\_data** (*list*) – Raw byte list of sent dns message response.
- **count** (*int*) – Question count.

**Return type** list (list of resource record/s)

**socket\_type** = 2

**static string** (*bytes: List[T]*)  
 Creates str from bytes.

**Parameters** **bytes** (*bytes*) – List of raw bytes.

**Return type** str

**static ui16** (*bytes: List[T]*)  
 Creates unsigned 16 bit int.

**Parameters** **bytes** (*list*) – List of raw bytes.

**Return type** unsigned 16 bit int

**static ui32** (*bytes: List[T]*)

Creates unsigned 32 bit int.

**Parameters** **bytes** (*list*) – List of raw bytes.

**Return type** unsigned 32 bit int

## 2.3 Enums

**class** `asyncdnsproxy.dnsproxy_enum.RecordType`

Bases: `enum.Enum`

DNS message and response record types.

`a` = 1

`aaaa` = 28

`afsdB` = 18

`apl` = 42

`caa` = 257

`cdnskey` = 60

`cds` = 59

`cert` = 37

`cname` = 5

`csync` = 62

`dhcid` = 49

`dlv` = 32769

`dname` = 39

`dnskey` = 48

`ds` = 43

`hip` = 55

`ipseckey` = 45

`key` = 25

`kx` = 36

`loc` = 29

`mx` = 15

`naptr` = 35

`ns` = 2

`nsec` = 47

`nsec3` = 50

`nsec3param` = 51

```

openpgpkey = 61
ptr = 12
rp = 17
rrsig = 46
sig = 24
smimea = 53
soa = 6
srv = 33
sshfp = 44
ta = 32768
tkey = 249
tlsa = 52
tsig = 250
txt = 16
uri = 256

```

```

class asyncdnsproxy.dnsproxy_enum.SocketType
    Bases: enum.Enum

    Socket types.

    tcp = 1
    udp = 2

```

## 2.4 Clients

```

class asyncdnsproxy.udp_client.UDPClient (address: str, port: int, blocking: bool = False, time-
                                         out: int = 10)

```

Bases: object

UDP client for connecting to a DNS server over UDP.

```

close ()
    Closes the socket connection.

```

```

>>> from asyncdnsproxy.udp_client import UDPClient
>>> udp_client = UDPClient('8.8.8.8', 53)
>>> udp_client.connect ()
>>> udp_client.close ()

```

```

connect ()
    Connects to given address and port.

```

```

>>> from asyncdnsproxy.udp_client import UDPClient
>>> udp_client = UDPClient('8.8.8.8', 53)
>>> udp_client.connect ()

```

```

receive (bufsize: int = 1024, flags: int = 0)
    Receives result of send data.

```

```
>>> from asyncdns.py.udp_client import UDPClient
>>> from asyncdns.py.dns_raw_message import DNSRawMessage
>>> from asyncdns.py.dns_enum import RecordType
```

```
>>> udp_client = UDPClient('8.8.8.8', 53)
>>> udp_client.connect()
>>> dns_raw_message = DNSRawMessage()
>>> data = dns_raw_message.query('google.com', RecordType.a)
>>> result = udp_client.send(data)
>>> response = udp_client.receive()
```

### Parameters

- **bufsize** (*int*) – Buffer size.
- **flags** (*int*) – Flags to send data.

**Return type** bytes

**send** (*data: bytes, flags: int = 0*)

Send bytes over UDP.

```
>>> from asyncdns.py.udp_client import UDPClient
>>> from asyncdns.py.dns_raw_message import DNSRawMessage
>>> from asyncdns.py.dns_enum import RecordType
```

```
>>> udp_client = UDPClient('8.8.8.8', 53)
>>> udp_client.connect()
>>> dns_raw_message = DNSRawMessage()
>>> data = dns_raw_message.query('google.com', RecordType.a)
>>> result = udp_client.send(data)
```

### Parameters

- **data** (*bytes*) – Bytes created by DNSRawMessage.
- **flags** (*int*) – Flags to send data.

**Return type** int

**class** `asyncdns.py.tcp_client.TCPClient` (*address: str, port: int, blocking: bool = False, timeout: int = 10*)

Bases: object

TCP client for connecting to a DNS server over TCP.

**close** ()

Closes the socket connection.

```
>>> from asyncdns.py.tcp_client import TCPClient
>>> tcp_client = TCPClient('8.8.8.8', 53)
>>> tcp_client.connect()
>>> tcp_client.close()
```

**connect** ()

Connects to given address and port.

```
>>> from asyncdnsproxy.tcp_client import TCPClient
>>> tcp_client = TCPClient('8.8.8.8', 53)
>>> tcp_client.connect()
```

**receive** (*bufsize: int = 1024, flags: int = 0*)

Receives result of send data.

```
>>> from asyncdnsproxy.tcp_client import TCPClient
>>> from asyncdnsproxy.dns_raw_message import DNSRawMessage
>>> from asyncdnsproxy.dnsproxy_enum import RecordType
```

```
>>> tcp_client = TCPClient('8.8.8.8', 53)
>>> tcp_client.connect()
>>> dns_raw_message = DNSRawMessage()
>>> data = dns_raw_message.query('google.com', RecordType.a, socket_
↳type=SocketType.tcp)
>>> result = tcp_client.send(data)
>>> response = tcp_client.receive()
```

#### Parameters

- **bufsize** (*int*) – Buffer size.
- **flags** (*int*) – Flags to send data.

**Return type** bytes

**send** (*data: bytes, flags: int = 0*)

Send bytes over UDP.

```
>>> from asyncdnsproxy.tcp_client import TCPClient
>>> from asyncdnsproxy.dns_raw_message import DNSRawMessage
>>> from asyncdnsproxy.dnsproxy_enum import RecordType
```

```
>>> tcp_client = TCPClient('8.8.8.8', 53)
>>> tcp_client.connect()
>>> dns_raw_message = DNSRawMessage()
>>> data = dns_raw_message.query('google.com', RecordType.a, socket_
↳type=SocketType.tcp)
>>> result = tcp_client.send(data)
```

#### Parameters

- **data** (*bytes*) – Bytes created by DNSRawMessage.
- **flags** (*int*) – Flags to send data.

**Return type** int

## CHAPTER 3

---

### Introduction

---

Pure Python asynchronous DNS client and server library. It works with Python 3.x.



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### a

`asyncdns.py.dns_message`, 5  
`asyncdns.py.dns_message_decoder`, 5  
`asyncdns.py.dns_message_header`, 4  
`asyncdns.py.dns_message_question`, 4  
`asyncdns.py.dns_message_resourcerecord`,  
4  
`asyncdns.py.dns_raw_message`, 3  
`asyncdns.py.dns.py_enum`, 7  
`asyncdns.py.tcp_client`, 9  
`asyncdns.py.udp_client`, 8



## A

asyncdns.py.dns.py\_enum.RecordType attribute), 7  
 aa (*asyncdns.py.dns\_raw\_message.Flag* attribute), 3  
 aaaa (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 ad (*asyncdns.py.dns\_raw\_message.Flag* attribute), 3  
 additional (*asyncdns.py.dns\_message.DNSMessage* attribute), 5  
 additional\_count (*asyncdns.py.dns\_message\_header.Header* attribute), 4  
 afsdb (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 answer\_count (*asyncdns.py.dns\_message\_header.Header* attribute), 4  
 answers (*asyncdns.py.dns\_message.DNSMessage* attribute), 5  
 apl (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 asyncdns.py.dns\_message (module), 5  
 asyncdns.py.dns\_message\_decoder (module), 5  
 asyncdns.py.dns\_message\_header (module), 4  
 asyncdns.py.dns\_message\_question (module), 4  
 asyncdns.py.dns\_message\_resourcerecord (module), 4  
 asyncdns.py.dns\_raw\_message (module), 3  
 asyncdns.py.dns.py\_enum (module), 7  
 asyncdns.py.tcp\_client (module), 9  
 asyncdns.py.udp\_client (module), 8  
 authority (*asyncdns.py.dns\_message.DNSMessage* attribute), 5  
 authority\_count (*asyncdns.py.dns\_message\_header.Header* attribute), 4

## C

caa (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 cd (*asyncdns.py.dns\_raw\_message.Flag* attribute), 3  
 cdnskey (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 cds (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 cert (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7

class\_type (*asyncdns.py.dns\_message\_question.Question* attribute), 4  
 class\_type (*asyncdns.py.dns\_message\_resourcerecord.ResourceRecord* attribute), 4  
 close () (*asyncdns.py.tcp\_client.TCPClient* method), 9  
 close () (*asyncdns.py.udp\_client.UDPClient* method), 8  
 cname (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 connect () (*asyncdns.py.tcp\_client.TCPClient* method), 9  
 connect () (*asyncdns.py.udp\_client.UDPClient* method), 8  
 csync (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7

## D

decode () (*asyncdns.py.dns\_message\_decoder.DNSMessageDecoder* static method), 5  
 decode\_name () (*asyncdns.py.dns\_message\_decoder.DNSMessageDecoder* static method), 6  
 dhcid (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 dlvs (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 dname (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 dnskey (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7  
 DNSMessage (class in *asyncdns.py.dns\_message*), 5  
 DNSMessageDecoder (class in *asyncdns.py.dns\_message\_decoder*), 5  
 DNSRawMessage (class in *asyncdns.py.dns\_raw\_message*), 3  
 ds (*asyncdns.py.dns.py\_enum.RecordType* attribute), 7

## F

Flag (class in *asyncdns.py.dns\_raw\_message*), 3  
 flags (*asyncdns.py.dns\_message\_header.Header* attribute), 4

## G

get\_ipv4\_address()  
(*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
method), 4

get\_ipv6\_address()  
(*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
method), 5

get\_txt()  
(*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
method), 5

## H

have\_answers (*asyncdnspy.dns\_raw\_message.State*  
attribute), 4

have\_flags (*asyncdnspy.dns\_raw\_message.State*  
attribute), 4

have\_header (*asyncdnspy.dns\_raw\_message.State*  
attribute), 4

have\_messageid (*asyncdnspy.dns\_raw\_message.State*  
attribute), 4

have\_questions (*asyncdnspy.dns\_raw\_message.State*  
attribute), 4

header (*asyncdnspy.dns\_message.DNSMessage* at-  
tribute), 5

Header (class in *asyncdnspy.dns\_message\_header*), 4

hip (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

## I

initial (*asyncdnspy.dns\_raw\_message.State* at-  
tribute), 4

ipseckey (*asyncdnspy.dnspy\_enum.RecordType*  
attribute), 7

## K

key (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

kx (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

## L

loc (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

## M

message\_id (*asyncdnspy.dns\_message\_header.Header*  
attribute), 4

mx (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

## N

name (*asyncdnspy.dns\_message\_question.Question* at-  
tribute), 4

name (*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
attribute), 5

naptr (*asyncdnspy.dnspy\_enum.RecordType* attribute),  
7

ns (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

nsec (*asyncdnspy.dnspy\_enum.RecordType* attribute), 7

nsec3 (*asyncdnspy.dnspy\_enum.RecordType* attribute),  
7

nsec3param (*asyncdnspy.dnspy\_enum.RecordType* at-  
tribute), 7

## O

offset (*asyncdnspy.dns\_message\_decoder.DNSMessageDecoder*  
attribute), 6

openpgpkey (*asyncdnspy.dnspy\_enum.RecordType* at-  
tribute), 7

## P

ptr (*asyncdnspy.dnspy\_enum.RecordType* attribute), 8

## Q

query() (*asyncdnspy.dns\_raw\_message.DNSRawMessage*  
method), 3

Question (class in *asyncdnspy.dns\_message\_question*),  
4

question\_count (*asyncdnspy.dns\_message\_header.Header*  
attribute), 4

question\_type (*asyncdnspy.dns\_message\_question.Question*  
attribute), 4

questions (*asyncdnspy.dns\_message.DNSMessage* at-  
tribute), 5

## R

ra (*asyncdnspy.dns\_raw\_message.Flag* attribute), 3

rd (*asyncdnspy.dns\_raw\_message.Flag* attribute), 3

read\_header() (*asyncdnspy.dns\_message\_decoder.DNSMessageDecoder*  
static method), 6

read\_questions() (*asyncdnspy.dns\_message\_decoder.DNSMessageDecoder*  
static method), 6

read\_resource\_records()  
(*asyncdnspy.dns\_message\_decoder.DNSMessageDecoder*  
static method), 6

receive() (*asyncdnspy.tcp\_client.TCPClient* method),  
10

receive() (*asyncdnspy.udp\_client.UDPClient*  
method), 8

record\_data (*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
attribute), 5

record\_data\_length  
(*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
attribute), 5

record\_type (*asyncdnspy.dns\_message\_resourcerecord.ResourceRecord*  
attribute), 5

RecordType (class in *asyncdnspy.dnspy\_enum*), 7

ResourceRecord (class in  
*asyncdnspy.dns\_message\_resourcerecord*),  
4

rp (*asyncdnspy.dnspy\_enum.RecordType* attribute), 8

rrsig (*asyncdnspy.dnspy\_enum.RecordType* attribute),  
8

## S

send() (*asyncdnsproxy.tcp\_client.TCPClient* method), 10  
 send() (*asyncdnsproxy.udp\_client.UDPClient* method), 9  
 sig (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 smimea (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 soa (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 socket\_type (*asyncdnsproxy.dns\_message\_decoder.DNSMessageDecoder* attribute), 6  
 SocketType (*class in asyncdnsproxy.dnsproxy\_enum*), 8  
 srv (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 sshfp (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 State (*class in asyncdnsproxy.dns\_raw\_message*), 4  
 string() (*asyncdnsproxy.dns\_message\_decoder.DNSMessageDecoder* static method), 6

## T

ta (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 tc (*asyncdnsproxy.dns\_raw\_message.Flag* attribute), 4  
 tcp (*asyncdnsproxy.dnsproxy\_enum.SocketType* attribute), 8  
 TCPClient (*class in asyncdnsproxy.tcp\_client*), 9  
 tkey (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 tlsa (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 tsig (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8  
 ttl (*asyncdnsproxy.dns\_message\_resourcerecord.ResourceRecord* attribute), 5  
 txt (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8

## U

udp (*asyncdnsproxy.dnsproxy\_enum.SocketType* attribute), 8  
 UDPClient (*class in asyncdnsproxy.udp\_client*), 8  
 ui16() (*asyncdnsproxy.dns\_message\_decoder.DNSMessageDecoder* static method), 6  
 ui32() (*asyncdnsproxy.dns\_message\_decoder.DNSMessageDecoder* static method), 7  
 uri (*asyncdnsproxy.dnsproxy\_enum.RecordType* attribute), 8