

---

# pingparsing Documentation

*Release 1.1.0*

**Tsuyoshi Hombashi**

**Aug 09, 2020**



# TABLE OF CONTENTS

<b>1 pingparsing</b>	<b>1</b>
1.1 Summary . . . . .	1
<b>2 Supported Environments</b>	<b>3</b>
2.1 Tested Environments . . . . .	3
<b>3 Premise</b>	<b>5</b>
<b>4 Installation</b>	<b>7</b>
<b>5 Dependencies</b>	<b>9</b>
5.1 Optional Dependencies . . . . .	9
<b>6 Docker Image</b>	<b>11</b>
<b>7 Usage</b>	<b>13</b>
7.1 CLI Usage . . . . .	13
7.1.1 Execute ping and parse the result . . . . .	13
7.1.2 Parse ping result file . . . . .	15
7.1.3 Parse from the standard input . . . . .	17
7.1.4 CLI help . . . . .	17
7.2 Library Usage . . . . .	19
7.2.1 Execute ping and parse the result . . . . .	19
7.2.2 Parsing ping command output . . . . .	19
<b>8 Reference</b>	<b>23</b>
8.1 Parser classes . . . . .	23
8.2 Transmitter classes . . . . .	25
8.3 Errors . . . . .	27
<b>9 Changelog</b>	<b>29</b>
<b>10 Sponsors</b>	<b>31</b>
<b>11 Indices and tables</b>	<b>33</b>
<b>12 Links</b>	<b>35</b>
<b>13 Indices and tables</b>	<b>37</b>
<b>Index</b>	<b>39</b>



## **PINGPARSING**

### **1.1 Summary**

pingparsing is a CLI-tool/Python-library parser and transmitter for ping command.



## SUPPORTED ENVIRONMENTS

- Linux
- Windows
- macOS

### 2.1 Tested Environments

OS	ping version
Ubuntu 16.04	iputils-ping 20121221-5ubuntu2
Ubuntu 18.04	iputils-ping 20161105-1ubuntu2
Debian 8.6	iputils-ping 20121221-5+b2
Fedora 25	iputils-20161105-1.fc25.x86_64
Windows 10	-
macOS 10.13	-





## PREMISE

`pingparsing` expects the locale at the `ping` command execution environment with English. Parsing the `ping` command output with any other locale may fail. This is because the output of the `ping` command will change depending on the locale setting.



## INSTALLATION

```
pip install pingarsing
```



## DEPENDENCIES

- Python 3.5+
- Python package dependencies (automatically installed)

### 5.1 Optional Dependencies

- **pingarsing[cli] extras**
  - **loguru**
    - \* Used for logging if the package installed
  - **Pygments**
    - \* Syntax highlighting to `pingarsing` command output when installed



## DOCKER IMAGE

[thombashi/pingparsing - Docker Hub](#)





## 7.1 CLI Usage

A CLI command (`pingparsing` command) included in the packaged. The command could do the following:

- Execute `ping` and parse the result
- **Parse ping results from:**
  - file(s)
  - the standard input

### 7.1.1 Execute ping and parse the result

If you specify destination(s) to the `pingparsing` command as positional arguments, the command executes `ping` for each destination(s) and parses the result. `ping` will execute in parallel for multiple destinations. The parsed result outputted with JSON format.

Listing 1: Execute with a single destination

```
$ pingparsing google.com
{
  "google.com": {
    "destination": "google.com",
    "packet_transmit": 10,
    "packet_receive": 10,
    "packet_loss_rate": 0.0,
    "packet_loss_count": 0,
    "rtt_min": 34.189,
    "rtt_avg": 46.054,
    "rtt_max": 63.246,
    "rtt_mdev": 9.122,
    "packet_duplicate_rate": 0.0,
    "packet_duplicate_count": 0
  }
}
```

Listing 2: Execute with multiple destinations

```
$ pingparsing google.com twitter.com
{
  "google.com": {
```

(continues on next page)

(continued from previous page)

```
"destination": "google.com",
"packet_transmit": 10,
"packet_receive": 10,
"packet_loss_rate": 0.0,
"packet_loss_count": 0,
"rtt_min": 37.341,
"rtt_avg": 44.538,
"rtt_max": 53.997,
"rtt_mdev": 5.827,
"packet_duplicate_rate": 0.0,
"packet_duplicate_count": 0
},
"twitter.com": {
  "destination": "twitter.com",
  "packet_transmit": 10,
  "packet_receive": 10,
  "packet_loss_rate": 0.0,
  "packet_loss_count": 0,
  "rtt_min": 45.377,
  "rtt_avg": 68.819,
  "rtt_max": 78.581,
  "rtt_mdev": 9.769,
  "packet_duplicate_rate": 0.0,
  "packet_duplicate_count": 0
}
}
```

Listing 3: Print ICMP packet replies

```
$ pingarsing google.com -c 3 --icmp-reply
{
  "google.com": {
    "destination": "google.com",
    "packet_transmit": 3,
    "packet_receive": 3,
    "packet_loss_count": 0,
    "packet_loss_rate": 0.0,
    "rtt_min": 19.885,
    "rtt_avg": 38.829,
    "rtt_max": 57.751,
    "rtt_mdev": 15.459,
    "packet_duplicate_count": 0,
    "packet_duplicate_rate": 0.0,
    "icmp_replies": [
      {
        "icmp_seq": 1,
        "ttl": 53,
        "time": 19.8,
        "duplicate": false
      },
      {
        "icmp_seq": 2,
        "ttl": 53,
        "time": 38.8,
        "duplicate": false
      }
    ],
  },
}
```

(continues on next page)

(continued from previous page)

```

    {
      "icmp_seq": 3,
      "ttl": 53,
      "time": 57.7,
      "duplicate": false
    }
  ]
}

```

## 7.1.2 Parse ping result file

### Input

```

$ cat ping.txt
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.

--- 192.168.0.1 ping statistics ---
1688 packets transmitted, 1553 received, +1 duplicates, 7% packet loss,
↪time 2987ms
rtt min/avg/max/mdev = 0.282/0.642/11.699/0.699 ms, pipe 2, ipg/ewma 1.
↪770/0.782 ms
$ cat osx.txt
PING google.com (172.217.6.238): 56 data bytes
64 bytes from 172.217.6.238: icmp_seq=0 ttl=53 time=20.482 ms
64 bytes from 172.217.6.238: icmp_seq=1 ttl=53 time=32.550 ms
64 bytes from 172.217.6.238: icmp_seq=2 ttl=53 time=32.013 ms
64 bytes from 172.217.6.238: icmp_seq=3 ttl=53 time=28.498 ms
64 bytes from 172.217.6.238: icmp_seq=4 ttl=53 time=46.093 ms

--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.482/31.927/46.093/8.292 ms

```

### Output

Listing 4: Parse multiple ping result files

```

$ pingarsing ping.txt osx.txt
{
  "osx.txt": {
    "destination": "google.com",
    "packet_transmit": 5,
    "packet_receive": 5,
    "packet_loss_rate": 0.0,
    "packet_loss_count": 0,
    "rtt_min": 20.482,
    "rtt_avg": 31.927,
    "rtt_max": 46.093,
    "rtt_mdev": 8.292,
    "packet_duplicate_rate": null,
    "packet_duplicate_count": null
  },
  "ping.txt": {
    "destination": "192.168.0.1",

```

(continues on next page)

(continued from previous page)

```
    "packet_transmit": 1688,  
    "packet_receive": 1553,  
    "packet_loss_rate": 7.997630331753558,  
    "packet_loss_count": 135,  
    "rtt_min": 0.282,  
    "rtt_avg": 0.642,  
    "rtt_max": 11.699,  
    "rtt_mdev": 0.699,  
    "packet_duplicate_rate": 0.0643915003219575,  
    "packet_duplicate_count": 1  
  }  
}
```

Listing 5: Print ICMP packet replies

```
$ pingarsing ping.txt osx.txt --icmp-reply  
{  
  "ping.txt": {  
    "destination": "google.com",  
    "packet_transmit": 60,  
    "packet_receive": 60,  
    "packet_loss_count": 0,  
    "packet_loss_rate": 0.0,  
    "rtt_min": 61.425,  
    "rtt_avg": 99.731,  
    "rtt_max": 212.597,  
    "rtt_mdev": 27.566,  
    "packet_duplicate_count": 0,  
    "packet_duplicate_rate": 0.0,  
    "icmp_replies": []  
  },  
  "osx.txt": {  
    "destination": "google.com",  
    "packet_transmit": 5,  
    "packet_receive": 5,  
    "packet_loss_count": 0,  
    "packet_loss_rate": 0.0,  
    "rtt_min": 20.482,  
    "rtt_avg": 31.927,  
    "rtt_max": 46.093,  
    "rtt_mdev": 8.292,  
    "packet_duplicate_count": 0,  
    "packet_duplicate_rate": 0.0,  
    "icmp_replies": [  
      {  
        "icmp_seq": 0,  
        "ttl": 53,  
        "time": 20.482,  
        "duplicate": false  
      },  
      {  
        "icmp_seq": 1,  
        "ttl": 53,  
        "time": 32.55,  
        "duplicate": false  
      }  
    ],  
  },  
}
```

(continues on next page)

(continued from previous page)

```

    {
      "icmp_seq": 2,
      "ttl": 53,
      "time": 32.013,
      "duplicate": false
    },
    {
      "icmp_seq": 3,
      "ttl": 53,
      "time": 28.498,
      "duplicate": false
    },
    {
      "icmp_seq": 4,
      "ttl": 53,
      "time": 46.093,
      "duplicate": false
    }
  ]
}

```

### 7.1.3 Parse from the standard input

```

$ ping -i 0.2 -w 20 192.168.2.101 | pingarsing -
{
  "destination": "192.168.2.101",
  "packet_transmit": 99,
  "packet_receive": 88,
  "packet_loss_count": 11,
  "packet_loss_rate": 11.11111111111111,
  "rtt_min": 1.615,
  "rtt_avg": 26.581,
  "rtt_max": 93.989,
  "rtt_mdev": 19.886,
  "packet_duplicate_count": 0,
  "packet_duplicate_rate": 0.0
}

```

### 7.1.4 CLI help

```

usage: pingarsing [-h] [-V] [--max-workers MAX_WORKERS]
                 [--timestamp {none,epoch,datetime}] [-c COUNT]
                 [-s PACKET_SIZE] [--ttl TTL] [-w DEADLINE]
                 [--timeout TIMEOUT] [-I INTERFACE] [--addopts OPTIONS]
                 [--indent INDENT] [--icmp-reply] [--no-color]
                 [--debug | --quiet]
                 destination_or_file [destination_or_file ...]

positional arguments:
  destination_or_file  Destinations to send ping, or files to parse. '-' for
                       parse the standard input.

```

(continues on next page)

## optional arguments:

```

-h, --help          show this help message and exit
-V, --version      show program's version number and exit
--max-workers MAX_WORKERS
                    Number of threads for when multiple destination/file
                    specified. defaults to equals to two times number of
                    cores.
--debug            for debug print.
--quiet           suppress execution log messages.

```

## Ping Options:

```

--timestamp {none,epoch,datetime}
                    [Only for LINUX] none: no timestamps. epoch: add
                    timestamps with UNIX epoch time format. datetime: add
                    timestamps with ISO time format.
-c COUNT, --count COUNT
                    Stop after sending the count. see also ping(8) [-c
                    count] option description.
-s PACKET_SIZE, --packet-size PACKET_SIZE
                    Specifies the number of data bytes to be sent.
--ttl TTL          Specifies the Time to Live.
-w DEADLINE, --deadline DEADLINE
                    Timeout before ping exits. valid time units are:
                    d/day/days, h/hour/hours, m/min/mins/minute/minutes,
                    s/sec/secs/second/seconds,
                    ms/msec/msecs/millisecond/milliseconds,
                    us/usec/usecs/microsecond/microseconds. if no unit
                    string found, considered seconds as the time unit. see
                    also ping(8) [-w deadline] option description. note:
                    meaning of the 'deadline' may differ system to system.
--timeout TIMEOUT
                    Time to wait for a response per packet. Valid time
                    units are: d/day/days, h/hour/hours,
                    m/min/mins/minute/minutes, s/sec/secs/second/seconds,
                    ms/msec/msecs/millisecond/milliseconds,
                    us/usec/usecs/microsecond/microseconds. if no unit
                    string found, considered milliseconds as the time
                    unit. Attempt to send packets with milliseconds
                    granularity in default. If the system does not support
                    timeout in milliseconds, round up as seconds. Use
                    system default if not specified. This option will be
                    ignored if the system does not support timeout itself.
                    See also ping(8) [-W timeout] option description.
                    note: meaning of the 'timeout' may differ system to
                    system.
-I INTERFACE, --interface INTERFACE
                    network interface
--adopts OPTIONS   extra command line options

```

## Output Options:

```

--indent INDENT    JSON output will be pretty-printed with the indent
                    level. (default= 4)
--icmp-reply, --icmp-replies
                    print results for each ICMP packet reply.
--no-color         Turn off colors.

```

Documentation: <https://pingarsing.rtf.d.io/>

(continues on next page)

(continued from previous page)

Issue tracker: <https://github.com/thombashi/pingarsing/issues>

## 7.2 Library Usage

### 7.2.1 Execute ping and parse the result

PingTransmitter class can execute ping command and obtain the ping output as a string.

#### Sample Code

```
import json
import pingarsing

ping_parser = pingarsing.PingParsing()
transmitter = pingarsing.PingTransmitter()
transmitter.destination = "google.com"
transmitter.count = 10
result = transmitter.ping()

print(json.dumps(ping_parser.parse(result).as_dict(), indent=4))
```

#### Output

```
{
  "destination": "google.com",
  "packet_transmit": 10,
  "packet_receive": 10,
  "packet_loss_rate": 0.0,
  "packet_loss_count": 0,
  "rtt_min": 34.458,
  "rtt_avg": 51.062,
  "rtt_max": 62.943,
  "rtt_mdev": 8.678,
  "packet_duplicate_rate": 0.0,
  "packet_duplicate_count": 0
}
```

### 7.2.2 Parsing ping command output

#### Sample Code

```
import json
from textwrap import dedent
import pingarsing

parser = pingarsing.PingParsing()
stats = parser.parse(dedent("""\
PING google.com (74.125.24.100) 56(84) bytes of data.
 [1524930937.003555] 64 bytes from 74.125.24.100: icmp_seq=1 ttl=39_
↔time=148 ms
 [1524930937.787175] 64 bytes from 74.125.24.100: icmp_seq=2 ttl=39_
↔time=137 ms
```

(continues on next page)

(continued from previous page)

```

[1524930938.787642] 64 bytes from 74.125.24.100: icmp_seq=3 ttl=39
↪time=137 ms
[1524930939.787653] 64 bytes from 74.125.24.100: icmp_seq=4 ttl=39
↪time=136 ms
[1524930940.788365] 64 bytes from 74.125.24.100: icmp_seq=5 ttl=39
↪time=136 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 136.537/139.174/148.006/4.425 ms
""")

print("[extract ping statistics]")
print(json.dumps(stats.as_dict(), indent=4))

print("\n[extract icmp replies]")
for icmp_reply in stats.icmp_replies:
    print(icmp_reply)

```

## Output

```

[extract ping statistics]
{
  "destination": "google.com",
  "packet_transmit": 5,
  "packet_receive": 5,
  "packet_loss_count": 0,
  "packet_loss_rate": 0.0,
  "rtt_min": 136.537,
  "rtt_avg": 139.174,
  "rtt_max": 148.006,
  "rtt_mdev": 4.425,
  "packet_duplicate_count": 0,
  "packet_duplicate_rate": 0.0
}

[extract icmp replies]
{'timestamp': datetime.datetime(2018, 4, 29, 0, 55, 37), 'icmp_seq': 1,
↪'ttl': 39, 'time': 148.0, 'duplicate': False}
{'timestamp': datetime.datetime(2018, 4, 29, 0, 55, 37), 'icmp_seq': 2,
↪'ttl': 39, 'time': 137.0, 'duplicate': False}
{'timestamp': datetime.datetime(2018, 4, 29, 0, 55, 38), 'icmp_seq': 3,
↪'ttl': 39, 'time': 137.0, 'duplicate': False}
{'timestamp': datetime.datetime(2018, 4, 29, 0, 55, 39), 'icmp_seq': 4,
↪'ttl': 39, 'time': 136.0, 'duplicate': False}
{'timestamp': datetime.datetime(2018, 4, 29, 0, 55, 40), 'icmp_seq': 5,
↪'ttl': 39, 'time': 136.0, 'duplicate': False}

```



## Recommended ping command execution

The following methods are recommended to execute `ping` command to get the output for parsing. These commands include an operation that changes the locale setting to English temporarily.

### Linux

```
LC_ALL=C ping <host or IP address> -w <seconds> [option] > <output.file>
```

### Windows

```
> chcp
Active code page: <XXX>    # get current code page

> chcp 437    # change code page to english
> ping <host or IP address> -n <ping count> > <output.file>
> chcp <XXX>    # restore code page
```

- **Reference**

- <https://technet.microsoft.com/en-us/library/cc733037>



## 8.1 Parser classes

**class** pingparsing.**PingParsing**

Parser class to parsing ping command output.

**parse** (*ping\_message*: Union[str, pingparsing.\_pingtransmitter.PingResult]) → pingparsing.\_stats.PingStats  
Parse ping command output.

**Parameters** *ping\_message* (str or *PingResult*) – ping command output.

**Returns** Parsed result.

**Return type** *PingStats*

**class** pingparsing.**PingStats** (\*args, \*\*kwargs)

**as\_dict** (*include\_icmp\_replies*: bool = False) → Dict[str, Optional[Union[str, int, float, Sequence[Dict[str, Union[bool, float, int, datetime.datetime]]]]]]  
ping statistics.

**Returns**

**Return type** dict

### Examples

```
>>> import pingparsing
>>> parser = pingparsing.PingParsing()
>>> parser.parse(ping_result)
>>> parser.as_dict()
{
    "destination": "google.com",
    "packet_transmit": 60,
    "packet_receive": 60,
    "packet_loss_rate": 0.0,
    "packet_loss_count": 0,
    "rtt_min": 61.425,
    "rtt_avg": 99.731,
    "rtt_max": 212.597,
    "rtt_mdev": 27.566,
    "packet_duplicate_rate": 0.0,
```

(continues on next page)

(continued from previous page)

```

    "packet_duplicate_count": 0
}

```

**as\_tuple()** → Tuple  
ping statistics.

**Returns**

**Return type** `namedtuple()`

## Examples

```

>>> import pingarsing
>>> parser = pingarsing.PingParsing()
>>> parser.parse(ping_result)
>>> parser.as_tuple()
PingResult(destination='google.com', packet_transmit=60, packet_receive=60,
↳ packet_loss_rate=0.0, packet_loss_count=0, rtt_min=61.425, rtt_avg=99.731,
↳ rtt_max=212.597, rtt_mdev=27.566, packet_duplicate_rate=0.0, packet_
↳ duplicate_count=0)

```

**property destination**

The ping destination.

**Returns**

**Return type** `str`

**property icmp\_replies**

ICMP packet reply information.

**Returns**

**Return type** `list of dict`

**is\_empty()**

**property packet\_duplicate\_count**

Number of duplicated packets.

**Returns** `None` when parsing Windows ping result.

**Return type** `int`

**property packet\_duplicate\_rate**

Percentage of duplicated packets [%].

**Returns** `None` if the value is not a number.

**Return type** `float`

**property packet\_loss\_count**

Number of packet losses.

**Returns** `None` if the value is not a number.

**Return type** `int`

**property packet\_loss\_rate**

Percentage of packet loss [%].

**Returns** `None` if the value is not a number.

**Return type** `float`

**property packet\_receive**

Number of packets received.

**Returns**

**Return type** `int`

**property packet\_transmit**

Number of packets transmitted.

**Returns**

**Return type** `int`

**property rtt\_avg**

Average round trip time of transmitted ICMP packets [msec].

**Returns**

**Return type** `float`

**property rtt\_max**

Maximum round trip time of transmitted ICMP packets [msec].

**Returns**

**Return type** `float`

**property rtt\_mdev**

Standard deviation of transmitted ICMP packets.

**Returns** `None` when parsing Windows ping result.

**Return type** `float`

**property rtt\_min**

Minimum round trip time of transmitted ICMP packets [msec].

**Returns**

**Return type** `float`

## 8.2 Transmitter classes

**class** `pingarsing.PingTransmitter`

Transmitter class to send ICMP packets by using the OS built-in ping command.

**count:** `Optional[int] = None`

Number of sending ICMP packets. This attribute ignored if the value is `None`.

**packet\_size:** `Optional[int] = None`

Specifies the number of data bytes to be sent.

**ttl:** `Optional[int] = None`

Specifies the Time to Live.

**ping\_option:** `str = ""`

Additional ping command option.

**interface:** `Optional[str] = None`

Interface name or zone-id. The attribute required when `destination` is IPv6 link-local scope address.

**timestamp:** `bool = False`

[Only for Linux environment] If `True`, add timestamp for each ping result. Defaults to `False`.

**auto\_codepage:** `bool = True`

[Only for Windows environment] Automatically change code page if `True`. Defaults to `True`.

**property deadline**

Timeout before ping exits. You can specify either a number or a string (e.g. "1sec"). If both `deadline` and `count` are `None`, If only a number is specified and a unit not found, the unit will be considered as seconds.

Unit	Available specifiers (str)
days	d/day/days
hours	h/hour/hours
minutes	m/min/mins/minute/minutes
seconds	s/sec/secs/second/seconds
milliseconds	ms/msec/msecs/millisecond/milliseconds
microseconds	us/usec/usecs/microsecond/microseconds

`deadline` automatically set to the default value (3 seconds). Defaults to `None`.

**Returns** `deadline`

**Return type** `humanreadable.Time`

**property destination**

Hostname or IP-address (IPv4/IPv6) to sending ICMP packets.

**property destination\_host**

Alias to `destination`

**ping()** → `pingparsing._pingtransmitter.PingResult`

Sending ICMP packets.

**Returns** `ping` command execution result.

**Return type** `PingResult`

**Raises** `ValueError` – If parameters not valid.

**property timeout**

Time to wait for a response per packet. You can specify either a number or a string (e.g. "1sec"). If only a number is specified and a unit not found, the unit will be considered as seconds.

Unit	Available specifiers (str)
days	d/day/days
hours	h/hour/hours
minutes	m/min/mins/minute/minutes
seconds	s/sec/secs/second/seconds
milliseconds	ms/msec/msecs/millisecond/milliseconds
microseconds	us/usec/usecs/microsecond/microseconds

Use system default timeout if the value is `None`. Defaults to `None`. If the system does not support timeout in milliseconds, round up as seconds.

**Returns** `timeout`

**Return type** `humanreadable.Time`

**class** pingarsing.**PingResult** (*stdout, stderr, returncode*)  
Data class to store ping command execution result.

**stdout**: **Optional[str]**  
Standard output of ping command execution result.

**stderr**: **Optional[str]**  
Standard error of ping command execution result.

**returncode**: **int**  
Return code of ping command execution result.

## 8.3 Errors

**exception** pingarsing.**ParseError** (*\*args, \*\*kwargs*)  
Bases: `Exception`

Exception raised when failed to parse ping results.

**class** pingarsing.error.**ParseErrorReason** (*value*)  
Bases: `enum.Enum`

An enumeration.

**HEADER\_NOT\_FOUND**

**EMPTY\_STATISTICS**





## CHANGELOG

<https://github.com/thombashi/pingparsing/releases>



**SPONSORS**

Become a sponsor



## INDICES AND TABLES

- `genindex`



## LINKS

- [GitHub repository](#)
- [Issue tracker](#)
- [pip](#): A tool for installing Python packages





**INDICES AND TABLES**

- search



## A

as\_dict() (*pingarsing.PingStats* method), 23  
 as\_tuple() (*pingarsing.PingStats* method), 24  
 auto\_codepage (*pingarsing.PingTransmitter* attribute), 26

## C

count (*pingarsing.PingTransmitter* attribute), 25

## D

deadline() (*pingarsing.PingTransmitter* property), 26  
 destination() (*pingarsing.PingStats* property), 24  
 destination() (*pingarsing.PingTransmitter* property), 26  
 destination\_host() (*pingarsing.PingTransmitter* property), 26

## E

EMPTY\_STATISTICS (*pingarsing.error.ParseErrorReason* attribute), 27

## H

HEADER\_NOT\_FOUND (*pingarsing.error.ParseErrorReason* attribute), 27

## I

icmp\_replies() (*pingarsing.PingStats* property), 24  
 interface (*pingarsing.PingTransmitter* attribute), 25  
 is\_empty() (*pingarsing.PingStats* method), 24

## P

packet\_duplicate\_count() (*pingarsing.PingStats* property), 24  
 packet\_duplicate\_rate() (*pingarsing.PingStats* property), 24  
 packet\_loss\_count() (*pingarsing.PingStats* property), 24  
 packet\_loss\_rate() (*pingarsing.PingStats* property), 24

packet\_receive() (*pingarsing.PingStats* property), 25  
 packet\_size (*pingarsing.PingTransmitter* attribute), 25  
 packet\_transmit() (*pingarsing.PingStats* property), 25  
 parse() (*pingarsing.PingParsing* method), 23  
 ParseError, 27  
 ParseErrorReason (*class in pingarsing.error*), 27  
 ping() (*pingarsing.PingTransmitter* method), 26  
 ping\_option (*pingarsing.PingTransmitter* attribute), 25  
 PingParsing (*class in pingarsing*), 23  
 PingResult (*class in pingarsing*), 26  
 PingStats (*class in pingarsing*), 23  
 PingTransmitter (*class in pingarsing*), 25

## R

returncode (*pingarsing.PingResult* attribute), 27  
 rtt\_avg() (*pingarsing.PingStats* property), 25  
 rtt\_max() (*pingarsing.PingStats* property), 25  
 rtt\_mdev() (*pingarsing.PingStats* property), 25  
 rtt\_min() (*pingarsing.PingStats* property), 25

## S

stderr (*pingarsing.PingResult* attribute), 27  
 stdout (*pingarsing.PingResult* attribute), 27

## T

timeout() (*pingarsing.PingTransmitter* property), 26  
 timestamp (*pingarsing.PingTransmitter* attribute), 25  
 ttl (*pingarsing.PingTransmitter* attribute), 25