

---

# **Python Client for Pilosa Documentation**

***Release 0.3.30***

**Pilosa Engineering**

**Aug 03, 2017**



---

## Contents:

---

<b>1</b>	<b>pilosa package</b>	<b>3</b>
1.1	Submodules . . . . .	3
1.2	pilosa.client module . . . . .	3
1.3	pilosa.exceptions module . . . . .	6
1.4	pilosa.orm module . . . . .	6
1.5	pilosa.response module . . . . .	9
1.6	pilosa.validator module . . . . .	10
1.7	pilosa.version module . . . . .	10
1.8	Module contents . . . . .	10
<b>2</b>	<b>Requirements</b>	<b>11</b>
<b>3</b>	<b>Install</b>	<b>13</b>
<b>4</b>	<b>Quick overview</b>	<b>15</b>
<b>5</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



Python client for [Pilosa](#) high performance distributed bitmap index.



## Submodules

### **pilosa.client module**

**class** `pilosa.client.Client` (*cluster\_or\_uri=None, connect\_timeout=30000, socket\_timeout=300000, pool\_size\_per\_route=10, pool\_size\_total=100, retry\_count=3*)

Bases: `object`

Pilosa HTTP client

This client uses Pilosa's http+protobuf API.

Usage:

```
import pilosa

# Create a Client instance
client = pilosa.Client()

# Create an Index instance
index = pilosa.Index("repository")

stargazer = index.frame("stargazer")
response = client.query(stargazer.bitmap(5))

# Act on the result
print(response.result)
```

- See [Pilosa API Reference](#).
- See [Query Language](#).

**create\_frame** (*frame*)

Creates a frame on the server using the given Frame object.

**Parameters** **frame** (*pilosa.Frame*) –

**Raises** **pilosa.FrameExistsError** – if there already is a frame with the given name

**create\_index** (*index*)

Creates an index on the server using the given Index object.

**Parameters** **index** (*pilosa.Index*) –

**Raises** **pilosa.IndexExistsError** – if there already is a index with the given name

**delete\_frame** (*frame*)

Deletes the given frame on the server.

**Parameters** **frame** (*pilosa.Frame*) –

**Raises** **pilosa.PilosaError** – if the frame does not exist

**delete\_index** (*index*)

Deletes the given index on the server.

**Parameters** **index** (*pilosa.Index*) –

**Raises** **pilosa.PilosaError** – if the index does not exist

**ensure\_frame** (*frame*)

Creates a frame on the server if it does not exist.

**Parameters** **frame** (*pilosa.Frame*) –

**ensure\_index** (*index*)

Creates an index on the server if it does not exist.

**Parameters** **index** (*pilosa.Index*) –

**import\_frame** (*frame, bit\_reader, batch\_size=100000*)

Imports a frame using the given bit reader

**Parameters**

- **frame** –
- **bit\_reader** –
- **batch\_size** –

**query** (*query, columns=False*)

Runs the given query against the server with the given options.

**Parameters**

- **query** (*pilosa.PqlQuery*) – a PqlQuery object with a non-null index
- **columns** (*bool*) – Enables returning column data from bitmap queries

**Returns** Pilosa response

**Return type** *pilosa.Response*

**schema** ()

**status** ()

**sync\_schema** (*schema*)



**class** pilosa.client.**Cluster** (\*hosts)

Contains hosts in a Pilosa cluster.

**Parameters** **hosts** – URIs of hosts. Leaving out hosts creates the default cluster

**add\_host** (uri)

Makes a host available.

**Parameters** **uri** (pilosa.URI) –

**copy** ()

**get\_host** ()

Returns the next host in the cluster.

**Returns** next host

**Return type** pilosa.URI

**remove\_host** (uri)

Makes a host unavailable.

**Parameters** **uri** (pilosa.URI) –

**class** pilosa.client.**URI** (scheme='http', host='localhost', port=10101)

Represents a Pilosa URI

A Pilosa URI consists of three parts:

- **Scheme**: Protocol of the URI. Default: http
- **Host**: Hostname or IP URI. Default: localhost
- **Port**: Port of the URI. Default 10101

All parts of the URI are optional. The following are equivalent:

- http://localhost:10101
- http://localhost
- http://:10101
- localhost:10101
- localhost
- :10101

**Parameters**

- **scheme** (str) – is the scheme of the Pilosa Server. Currently only http is supported
- **host** (str) – is the hostname or IP address of the Pilosa server
- **port** (int) – is the port of the Pilosa server

**classmethod** **address** (address)

Creates a URI from an address.

**Parameters** **address** (str) – of the form \${SCHEME}://\${HOST}:{PORT}

**Returns** a Pilosa URI

**Type** pilosa.URI

## **pilosa.exceptions module**

**exception** `pilosa.exceptions.PilosaError`

Bases: `exceptions.Exception`

**exception** `pilosa.exceptions.ValidationError`

Bases: `pilosa.exceptions.PilosaError`

**exception** `pilosa.exceptions.PilosaURIError`

Bases: `pilosa.exceptions.PilosaError`

**exception** `pilosa.exceptions.IndexExistsError`

Bases: `pilosa.exceptions.PilosaError`

**exception** `pilosa.exceptions.FrameExistsError`

Bases: `pilosa.exceptions.PilosaError`

## **pilosa.orm module**

**class** `pilosa.orm.TimeQuantum(value)`

Valid time quantum values for frames having support for that.

•See: [Data Model](#)

**DAY** = `<pilosa.orm.TimeQuantum instance>`

**DAY\_HOUR** = `<pilosa.orm.TimeQuantum instance>`

**HOURL** = `<pilosa.orm.TimeQuantum instance>`

**MONTH** = `<pilosa.orm.TimeQuantum instance>`

**MONTH\_DAY** = `<pilosa.orm.TimeQuantum instance>`

**MONTH\_DAY\_HOUR** = `<pilosa.orm.TimeQuantum instance>`

**NONE** = `<pilosa.orm.TimeQuantum instance>`

**YEAR** = `<pilosa.orm.TimeQuantum instance>`

**YEAR\_MONTH** = `<pilosa.orm.TimeQuantum instance>`

**YEAR\_MONTH\_DAY** = `<pilosa.orm.TimeQuantum instance>`

**YEAR\_MONTH\_DAY\_HOUR** = `<pilosa.orm.TimeQuantum instance>`

**class** `pilosa.orm.CacheType(value)`

**DEFAULT** = `<pilosa.orm.CacheType instance>`

**LRU** = `<pilosa.orm.CacheType instance>`

**RANKED** = `<pilosa.orm.CacheType instance>`

**class** `pilosa.orm.Schema`

Schema is a container for index objects

**index** (*name*, *column\_label*=*'columnID'*, *time\_quantum*=`<pilosa.orm.TimeQuantum instance>`)

Returns an index object with the given name and options.

If the index didn't exist in the schema, it is added to the schema.

**Parameters**

- **name** (*str*) – index name
- **column\_label** (*str*) – a valid column label
- **time\_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum

**Returns** Index object

- See [Data Model](#)
- See [Query Language](#)

**class** `pilosa.orm.Index` (*name*, *column\_label*=*'columnID'*, *time\_quantum*=<*pilosa.orm.TimeQuantum* instance>)

The purpose of the Index is to represent a data namespace.

You cannot perform cross-index queries. Column-level attributes are global to the Index.

**Parameters**

- **name** (*str*) – index name
- **column\_label** (*str*) – a valid column label
- **time\_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum

- See [Data Model](#)
- See [Query Language](#)

**batch\_query** (\**queries*)

Creates a batch query.

**Parameters** **queries** (*pilosa.PQLQuery*) – the queries in the batch

**Returns** Pilosa batch query

**Return type** `pilosa.PQLBatchQuery`

**copy** (*frames=True*)

**count** (*bitmap*)

Creates a Count query.

Count returns the number of set bits in the BITMAP\_CALL passed in.

**Parameters** **bitmap** (*pilosa.PQLQuery*) – the bitmap query

**Returns** Pilosa query

**Return type** `pilosa.PQLQuery`

**difference** (\**bitmaps*)

Creates a Difference query.

Difference returns all of the bits from the first BITMAP\_CALL argument passed to it, without the bits from each subsequent BITMAP\_CALL.

**Parameters** **bitmaps** (*pilosa.PQLBitmapQuery*) – 0 or more bitmap queries to differentiate

**Returns** Pilosa bitmap query

**Return type** `pilosa.PQLBitmapQuery`

**Raises** `PilosaError` – if the number of bitmaps is less than 1

**frame** (*name*, *row\_label*=*'rowID'*, *time\_quantum*=<*pilosa.orm.TimeQuantum* *instance*>, *inverse\_enabled*=*False*, *cache\_type*=<*pilosa.orm.CacheType* *instance*>, *cache\_size*=0)  
Creates a frame object with the specified name and defaults.

**Parameters**

- **name** (*str*) – frame name
- **row\_label** (*str*) – a valid row label
- **time\_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum for the frame. If a Frame has a time quantum, then Views are generated for each of the defined time segments.
- **inverse\_enabled** (*bool*) –
- **cache\_type** (*pilosa.CacheType*) – *CacheType.DEFAULT*, *CacheType.LRU* or *CacheType.RANKED*
- **cache\_size** (*int*) – Values greater than 0 sets the cache size. Otherwise uses the default cache size

**Returns** Pilosa frame

**Return type** *pilosa.Frame*

**intersect** (\**bitmaps*)

Creates an Intersect query.

Intersect performs a logical AND on the results of each BITMAP\_CALL query passed to it.

**Parameters** **bitmaps** (*pilosa.PQLBitmapQuery*) – 1 or more bitmap queries to intersect

**Returns** Pilosa bitmap query

**Return type** *pilosa.PQLBitmapQuery*

**Raises** *PilosaError* – if the number of bitmaps is less than 1

**raw\_query** (*query*)

Creates a raw query.

Note that the query is not validated before sending to the server.

**Parameters** **query** (*str*) –

**Returns** Pilosa query

**Return type** *pilosa.PQLQuery*

**set\_column\_attrs** (*column\_id*, *attrs*)

Creates a SetColumnAttrs query.

SetColumnAttrs associates arbitrary key/value pairs with a column in an index.

Following object types are accepted:

- *int*
- *str*
- *bool*
- *float*

**Parameters**

- **column\_id** (*int*) –
- **attrs** (*dict*) – column attributes

**Returns** Pilosa query

**Return type** pilosa.PQLQuery

**union** (\**bitmaps*)

Creates a Union query.

Union performs a logical OR on the results of each BITMAP\_CALL query passed to it.

**Parameters** **bitmaps** (*pilosa.PQLBitmapQuery*) – 0 or more bitmap queries to union

**Returns** Pilosa bitmap query

**Return type** pilosa.PQLBitmapQuery

**class** pilosa.orm.**PQLQuery** (*pql, index*)

**serialize** ()

**class** pilosa.orm.**PQLBatchQuery** (*index*)

**add** (\**queries*)

**serialize** ()

## pilosa.response module

**class** pilosa.response.**BitmapResult** (*bits=None, attributes=None*)

Represents a result from Bitmap, Union, Intersect, Difference and Range PQL calls.

•See [Query Language](#)

**classmethod** **from\_internal** (*obj*)

**class** pilosa.response.**CountResultItem** (*id, count*)

Represents a result from TopN call.

•See [Query Language](#)

**class** pilosa.response.**QueryResult** (*bitmap=None, count\_items=None, count=0*)

Represent one of the results in the response.

•See [Query Language](#)

**classmethod** **from\_internal** (*obj*)

**class** pilosa.response.**ColumnItem** (*id, attributes*)

Contains data about a column.

Column data is returned from `QueryResponse.getColumns()` method. They are only returned if `Client.query` was called with `columns=True`.

**class** pilosa.response.**QueryResponse** (*results=None, columns=None, error\_message=''*)

Bases: object

Represents the response from a Pilosa query.

•See [Query Language](#)

**column**

**result**

## **pilosa.validator module**

```
pilosa.validator.valid_index_name(index_name)
pilosa.validator.validate_index_name(index_name)
pilosa.validator.valid_frame_name(frame_name)
pilosa.validator.validate_frame_name(frame_name)
pilosa.validator.valid_label(label)
pilosa.validator.validate_label(label)
```

## **pilosa.version module**

```
pilosa.version.get_version()
    Returns the version being used
```

## **Module contents**

## CHAPTER 2

---

### Requirements

---

- Python 2.6 and higher or Python 3.3 and higher





## CHAPTER 3

---

### Install

---

Pilosa client is on [PyPI](#). You can install the library using `pip`:

```
pip install pilosa
```



## CHAPTER 4

---

### Quick overview

---

Assuming Pilosa server is running at `localhost:10101` (the default):

```
import pilosa

# Create the default client
client = pilosa.Client()

# Create an Index object
myindex = pilosa.Index("myindex")

# Make sure the index exists on the server
client.ensure_index(myindex)

# Create a Frame object
myframe = myindex.frame("myframe")

# Make sure the frame exists on the server
client.ensure_frame(myframe)

# Send a SetBit query. PilosaError is thrown if execution of the query fails.
client.query(myframe.setbit(5, 42))

# Send a Bitmap query. PilosaError is thrown if execution of the query fails.
response = client.query(myframe.bitmap(5))

# Get the result
result = response.result

# Act on the result
if result:
    bits = result.bitmap.bits
    print("Got bits: ", bits)

# You can batch queries to improve throughput
response = client.query(
```

```
    myindex.batch_query(  
        myframe.bitmap(5),  
        myframe.bitmap(10),  
    )  
)  
for result in response.results:  
    # Act on the result  
    print(result)
```

## CHAPTER 5

---

### Indices and tables

---

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



### p

- `pilosa`, [10](#)
- `pilosa.client`, [3](#)
- `pilosa.exceptions`, [6](#)
- `pilosa.orm`, [6](#)
- `pilosa.response`, [9](#)
- `pilosa.validator`, [10](#)
- `pilosa.version`, [10](#)





## A

`add()` (pilosa.orm.PQLBatchQuery method), 9  
`add_host()` (pilosa.client.Cluster method), 5  
`address()` (pilosa.client.URI class method), 5

## B

`batch_query()` (pilosa.orm.Index method), 7  
`BitmapResult` (class in pilosa.response), 9

## C

`CacheType` (class in pilosa.orm), 6  
`Client` (class in pilosa.client), 3  
`Cluster` (class in pilosa.client), 4  
`column` (pilosa.response.QueryResponse attribute), 9  
`ColumnItem` (class in pilosa.response), 9  
`copy()` (pilosa.client.Cluster method), 5  
`copy()` (pilosa.orm.Index method), 7  
`count()` (pilosa.orm.Index method), 7  
`CountResultItem` (class in pilosa.response), 9  
`create_frame()` (pilosa.client.Client method), 3  
`create_index()` (pilosa.client.Client method), 4

## D

`DAY` (pilosa.orm.TimeQuantum attribute), 6  
`DAY_HOUR` (pilosa.orm.TimeQuantum attribute), 6  
`DEFAULT` (pilosa.orm.CacheType attribute), 6  
`delete_frame()` (pilosa.client.Client method), 4  
`delete_index()` (pilosa.client.Client method), 4  
`difference()` (pilosa.orm.Index method), 7

## E

`ensure_frame()` (pilosa.client.Client method), 4  
`ensure_index()` (pilosa.client.Client method), 4

## F

`frame()` (pilosa.orm.Index method), 7  
`FrameExistsError`, 6  
`from_internal()` (pilosa.response.BitmapResult class method), 9

`from_internal()` (pilosa.response.QueryResult class method), 9

## G

`get_host()` (pilosa.client.Cluster method), 5  
`get_version()` (in module pilosa.version), 10

## H

`HOURL` (pilosa.orm.TimeQuantum attribute), 6

## I

`import_frame()` (pilosa.client.Client method), 4  
`Index` (class in pilosa.orm), 7  
`index()` (pilosa.orm.Schema method), 6  
`IndexExistsError`, 6  
`intersect()` (pilosa.orm.Index method), 8

## L

`LRU` (pilosa.orm.CacheType attribute), 6

## M

`MONTH` (pilosa.orm.TimeQuantum attribute), 6  
`MONTH_DAY` (pilosa.orm.TimeQuantum attribute), 6  
`MONTH_DAY_HOUR` (pilosa.orm.TimeQuantum attribute), 6

## N

`NONE` (pilosa.orm.TimeQuantum attribute), 6

## P

`pilosa` (module), 10  
`pilosa.client` (module), 3  
`pilosa.exceptions` (module), 6  
`pilosa.orm` (module), 6  
`pilosa.response` (module), 9  
`pilosa.validator` (module), 10  
`pilosa.version` (module), 10  
`PilosaError`, 6  
`PilosaURIError`, 6

PQLBatchQuery (class in pilosa.orm), 9

PQLQuery (class in pilosa.orm), 9

## Q

query() (pilosa.client.Client method), 4

QueryResponse (class in pilosa.response), 9

QueryResult (class in pilosa.response), 9

## R

RANKED (pilosa.orm.CacheType attribute), 6

raw\_query() (pilosa.orm.Index method), 8

remove\_host() (pilosa.client.Cluster method), 5

result (pilosa.response.QueryResponse attribute), 9

## S

Schema (class in pilosa.orm), 6

schema() (pilosa.client.Client method), 4

serialize() (pilosa.orm.PQLBatchQuery method), 9

serialize() (pilosa.orm.PQLQuery method), 9

set\_column\_attrs() (pilosa.orm.Index method), 8

status() (pilosa.client.Client method), 4

sync\_schema() (pilosa.client.Client method), 4

## T

TimeQuantum (class in pilosa.orm), 6

## U

union() (pilosa.orm.Index method), 9

URI (class in pilosa.client), 5

## V

valid\_frame\_name() (in module pilosa.validator), 10

valid\_index\_name() (in module pilosa.validator), 10

valid\_label() (in module pilosa.validator), 10

validate\_frame\_name() (in module pilosa.validator), 10

validate\_index\_name() (in module pilosa.validator), 10

validate\_label() (in module pilosa.validator), 10

ValidationError, 6

## Y

YEAR (pilosa.orm.TimeQuantum attribute), 6

YEAR\_MONTH (pilosa.orm.TimeQuantum attribute), 6

YEAR\_MONTH\_DAY (pilosa.orm.TimeQuantum attribute), 6

YEAR\_MONTH\_DAY\_HOUR (pilosa.orm.TimeQuantum attribute), 6