aiozk

Release 0.28.0

unknown

Dec 17, 2020

CONTENTS

1 Installation **3**

2 Getting Started **5**
2.1 Making use of recipes 5

3 Source code **7**

4 Authors **9**
4.1 API References 9
4.2 Recipe API References 17
4.3 Contributing 18

5 Indices and tables **19**

Python Module Index **21**

Index **23**

Asynchronous Zookeeper Client for *asyncio* and Python.

Current version is 0.28.0.

CHAPTER

ONE

INSTALLATION

```
$ pip install aiozk
```

CHAPTER

TWO

GETTING STARTED

Here's a python code for creating a znode and then getting data from the znode.

```
import aiozk
import asyncio


async def main():
    zk = aiozk.ZKClient('server1:2181,server2:2181,server3:2181')
    await zk.start()
    await zk.ensure_path('/greeting/to')
    await zk.create('/greeting/to/world', 'hello world')
    data = await zk.get_data('/greeting/to/world')
    print(data)
    # b'hello world' is printed
    await zk.delete('/greeting/to/world')
    await zk.close()


asyncio.run(main())
```

2.1 Making use of recipes

Recipe objects can be created via .recipes attribute. Supported recipes are as follows: `DataWatcher`, `ChildrenWatcher`, `Lock`, `SharedLock`, `Lease`, `Barrier`, `DoubleBarrier`, `LeaderElection`, `Party`, `Counter`, `TreeCache`, `Allocator`

```
import aiozk
import asyncio


async def main():
    zk = aiozk.ZKClient('localhost:2181')
    await zk.start()
    lock = zk.recipes.Lock('/path/to/lock')
    async with await lock.acquire():
        print('do some critical stuff')
    await zk.close()


asyncio.run(main())
```

CHAPTER

THREE

SOURCE CODE

The project is hosted on Github

CHAPTER

FOUR

AUTHORS

aiozk is written mostly by Kirill Pinchuk, Junyeong Jeong and several contributors.

4.1 API References

4.1.1 ZKClient

class aiozk.**ZKClient**(*servers*, *chroot=None*, *session_timeout=10*, *default_acl=None*, *retry_policy=None*, *allow_read_only=False*, *read_timeout=None*, *loop=None*)

The class of Zookeeper Client

Parameters

- **servers** (*str*) – Server list to which ZKClient tries connecting. Specify a comma (,) separated server list. A server is defined as address:port format.
- **chroot** (*str*) – Root znode path inside Zookeeper data hierarchy.
- **session_timeout** (*float*) – Zookeeper session timeout.
- **default_acl** (aiozk.ACL) – Default ACL for .create and .ensure_path coroutines. If acl parameter of them is not passed this ACL is used instead. If None is passed for default_acl, then ACL for unrestricted access is applied. This means that scheme is world, id is anyone and all READ/WRITE/CREATE/DELETE/ADMIN permission bits will be set to the new znode.
- **retry_policy** (aiozk.RetryPolicy) – Retry policy. If None, RetryPolicy.forever() is used instead.
- **allow_read_only** (*bool*) – True if you allow this client to make use of read only Zookeeper session, otherwise False.
- **read_timeout** (*float*) – Timeout on reading from Zookeeper server in seconds.
- **loop** – event loop. If None, asyncio.get_event_loop() is used.

async start()

Start Zookeeper session and await for session connected.

async close()

Close Zookeeper session and await for session closed.

async exists(*path*, *watch=False*)

Check whether the path exists.

Parameters

- **path** (*str*) – Path of znode
- **watch** (*bool*) – If True, a watch is set as a side effect

Returns True if it exists otherwise False

Return type bool

async create(*path*, *data=None*, *acl=None*, *ephemeral=False*, *sequential=False*, *container=False*)
Create a znode at the path.

Parameters

- **path** (*str*) – Path of znode
- **data** (*str, bytes*) – Data which will be stored at the znode if the request succeeds
- **acl** (aiozk.ACL) – ACL to be set to the new znode.
- **ephemeral** (*bool*) – True for creating ephemeral znode otherwise False
- **sequential** (*bool*) – True for creating seqeuence znode otherwise False
- **container** (*bool*) – True for creating container znode otherwise False

Returns Path of the created znode

Return type str

Raises ***aiozk.exc.NodeExists*** – Can be raised if a znode at the same path already exists.

async ensure_path(*path*, *acl=None*)
Ensure all znodes exist in the path. Missing Znodes will be created.

Parameters

- **path** (*str*) – Path of znode
- **acl** (aiozk.ACL) – ACL to be set to the new znodes

async delete(*path*, *force=False*)
Delete a znode at the path.

Parameters

- **path** (*str*) – Path of znode
- **force** (*bool*) – True for ignoring version of the znode. A version of a znode is used as an optimistic lock mechanism. Set false for making use of a version that is tracked by a stat cache of ZKClient.

Raises ***aiozk.exc.NoNode*** – Raised if path does not exist.

async deleteall(*path*)
Delete all znodes in the path recursively.

Parameters **path** (*str*) – Path of znode

Raises ***aiozk.exc.NoNode*** – Raised if path does not exist.

async get(*path*, *watch=False*)
Get data as bytes and stat of znode.

Parameters

- **path** (*str*) – Path of znode
- **watch** (*bool*) – True for setting a watch event as a side effect, otherwise False

Returns Data and stat of znode

Return type (bytes, *aiozk.protocol.stat.Stat*)

Raises **aiozk.exc.NoNode** – Can be raised if path does not exist

async get_data(*path*, *watch=False*)
Get data as bytes.

Parameters

- **path** (*str*) – Path of znode
- **watch** (*bool*) – True for setting a watch event as a side effect, otherwise False

Returns Data

Return type bytes

Raises **aiozk.exc.NoNode** – Can be raised if path does not exist

async set(*path*, *data*, *version*)
Set data to znode. Prefer using .set_data than this method unless you have to control the concurrency.

Parameters

- **path** (*str*) – Path of znode
- **data** (*str or bytes*) – Data to store at znode
- **version** (*int*) – Version of znode data to be modified.

Returns Response stat

Return type *aiozk.protocol.stat.Stat*

Raises

- **aiozk.exc.NoNode** – Raised if znode does not exist
- **aiozk.exc.BadVersion** – Raised if version does not match the actual version of the data. The update failed.

async set_data(*path*, *data*, *force=False*)
Set data to znode without needing to handle version.

Parameters

- **path** (*str*) – Path of znode
- **data** (*bytes or str*) – Data to be stored at znode
- **force** (*bool*) – True for ignoring data version. False for using version from stat cache.

Raises

- **aiozk.exc.NoNode** – Raised if znode does not exist
- **aiozk.exc.BadVersion** – Raised if force parameter is False and only if supplied version from stat cache does not match the actual version of znode data.

async get_children(*path*, *watch=False*)
Get all children names. Returned names are only basename and they does not include dirname.

Parameters

- **path** (*str*) – Path of znode
- **watch** (*bool*) – True for setting a watch event as a side effect otherwise False

Returns Names of children znodes

Return type [str]

Raises ***aiozk.exc.NoNode*** – Raised if znode does not exist

async get_acl(*path*)
Get list of ACLs associated with the znode

Parameters **path** (*str*) – Path of znode

Returns List of ACLs associated with the znode

Return type [*aiozk.ACL*]

async set_acl(*path*, *acl*, *force=False*)
Set ACL to the znode.

Parameters

- **path** (*str*) – Path of znode
- **acl** (aiozk.ACL) – ACL for the znode
- **force** (*bool*) – True for ignoring ACL version of the znode when setting ACL to the actual znode. False for using ACL version from the stat cache.

Raises

- ***aiozk.exc.NoNode*** – Raised if znode does not exist
- ***aiozk.exc.BadVersion*** – Raised if force parameter is False and only if the supplied version from stat cache does not match the actual ACL version of the znode.

begin_transaction()
Return Transaction instance which provides methods for read/write operations and commit method. This instance is used for transaction request.

Returns Transaction instance which can be used for adding read/write operations

Return type *aiozk.transaction.Transaction*

4.1.2 Transaction

class aiozk.transaction.**Transaction**(*client*)
Transaction request builder

Parameters **client** (aiozk.ZKClient) – Client instance

check_version(*path*, *version*)
Check znode version

Parameters

- **path** (*str*) – Znode path
- **version** (*int*) – Znode version

Returns None

coroutine commit()
Send all calls in transaction request and return results

Returns Transaction results

Return type *aiozk.transaction.Result*

Raises **ValueError** – On no operations to commit

create(*path*, *data=None*, *acl=None*, *ephemeral=False*, *sequential=False*, *container=False*)
Create new znode

Parameters

- **path** (*str*) – Znode path
- **data** (*str or bytes*) – Data to store in node
- **acl** (*[aiozk.ACL]*) – List of ACLs
- **ephemeral** (*bool*) – Ephemeral node type
- **sequential** (*bool*) – Sequential node type
- **container** (*bool*) – Container node type

Returns None

Raises **ValueError** – when *containers* feature is not supported by Zookeeper server (< 3.5.1)

delete(*path*, *version=- 1*)
Delete znode

Parameters

- **path** (*str*) – Znode path
- **version** (*int*) – Current version of node

Returns None

set_data(*path*, *data*, *version=- 1*)
Set data to znode

Parameters

- **path** (*str*) – Znode path
- **data** (*str or bytes*) – Data to store in node
- **version** (*int*) – Current version of node

Returns None

class aiozk.transaction.**Result**
Transaction result aggregator

Contains attributes:

- **checked** Set with results of `check_version()` methods
- **created** Set with results of `create()` methods
- **updated** Set with results of `set_data()` methods
- **deleted** Set with results of `delete()` methods

4.1.3 ACL

class aiozk.**ACL**(***kwargs*)

ACL object. Used to control access to its znodes.

Do not create this object directly, use `aiozk.ACL.make()` instead.

classmethod make(*scheme*, *id*, *read=False*, *write=False*, *create=False*, *delete=False*, *admin=False*)

Create ACL

Parameters

- **scheme** (*str*) – ACL scheme, one of following:
  - **world** has a single id, anyone, that represents **anyone**.
  - **auth** doesn't use any id, represents any authenticated user.
  - **digest** uses a username:password string to generate MD5 hash which is then used as an ACL ID identity.
  - **host** uses the client host name as an ACL ID identity.
  - **ip** uses the client host IP or CIDR as an ACL ID identity.
- **id** (*str*) – ACL ID identity.
- **read** (*bool*) – Permission, can get data from a node and list its children
- **write** (*bool*) – Permission, can set data for a node
- **create** (*bool*) – Permission, can create a child node
- **delete** (*bool*) – Permission, can delete a child node
- **admin** (*bool*) – Permission, can set permissions

Returns ACL instance

Return type *aiozk.ACL*

4.1.4 Stat

class aiozk.protocol.stat.**Stat**(***kwargs*)

Znode stat structure

Contains attributes:

- **created_zxid** The zxid of the change that created this znode.
- **last_modified_zxid** The zxid of the change that last modified this znode.
- **created** The time in milliseconds from epoch when this znode was created.
- **modified** The time in milliseconds from epoch when this znode was last modified.
- **version** The number of changes to the data of this znode.
- **child_version** The number of changes to the children of this znode.
- **acl_version** The number of changes to the ACL of this znode.
- **ephemeral_owner** The session id of the owner of this znode if the znode is an ephemeral node. If it is not an ephemeral node, it will be zero.
- **data_length** The length of the data field of this znode.

- **num_children** The number of children of this znode.
- **last_modified_children** The zxid of the change that last modified this znode children.

4.1.5 RetryPolicy

class aiozk.**RetryPolicy**(*try_limit*, *sleep_func*)

The class of Retry policy enforcer for Zookeper calls.

Class methods of this class implement several different retry policies.

Custom implementation can be provided using constructor directly with following parameters:

Parameters

- **try_limit** (*int*) – Retry attempts limit
- **sleep_func** (*func*) – function that calculates sleep time. Accepts one parameter, list of timestamps of each attempt

classmethod once()

One retry policy

Returns Rolicy with one retry

Return type *aiozk.RetryPolicy*

classmethod n_times(*n*)

n times retry policy, **no delay between retries**

Parameters **n** – retries limit

Returns Policy with *n* retries

Return type *aiozk.RetryPolicy*

classmethod forever()

Forever retry policy, **no delay between retries**

Returns Retry forever policy

Return type *aiozk.RetryPolicy*

classmethod exponential_backoff(*base=2*, *maximum=None*)

Exponential backoff retry policy.

Parameters

- **base** – base of exponentiation
- **maximum** – optional timeout in seconds

Returns Exponential backoff policy

Return type *aiozk.RetryPolicy*

classmethod until_elapsed(*timeout*)

Retry until *timeout* elapsed policy

Parameters **timeout** – retry time delay

Returns Retry until elapsed policy.

Return type *aiozk.RetryPolicy*

4.1.6 Exceptions

exception aiozk.exc.**APIError**

exception aiozk.exc.**AuthFailed**

exception aiozk.exc.**BadArguments**

exception aiozk.exc.**BadVersion**

exception aiozk.exc.**ConnectError**(*host*, *port*, *server_id=None*)

exception aiozk.exc.**ConnectionLoss**

exception aiozk.exc.**DataInconsistency**

exception aiozk.exc.**EphemeralOnLocalSession**

exception aiozk.exc.**FailedRetry**

exception aiozk.exc.**InvalidACL**

exception aiozk.exc.**InvalidCallback**

exception aiozk.exc.**InvalidClientState**

exception aiozk.exc.**InvalidStateTransition**

exception aiozk.exc.**MarshallingError**

exception aiozk.exc.**NewConfigNoQuorum**

exception aiozk.exc.**NoAuth**

exception aiozk.exc.**NoChildrenForEphemerals**

exception aiozk.exc.**NoNode**

exception aiozk.exc.**NoServersError**

exception aiozk.exc.**NoWatcher**

exception aiozk.exc.**NodeExists**

exception aiozk.exc.**NotEmpty**

exception aiozk.exc.**NotReadOnly**

exception aiozk.exc.**OperationTimeout**

exception aiozk.exc.**ReconfigInProcess**

exception aiozk.exc.**ResponseError**

exception aiozk.exc.**RolledBack**

exception aiozk.exc.**RuntimeInconsistency**

exception aiozk.exc.**SessionExpired**

exception aiozk.exc.**SessionLost**

exception aiozk.exc.**SessionMoved**

exception aiozk.exc.**TimeoutError**

exception aiozk.exc.**TransactionFailed**

exception aiozk.exc.**UnfinishedRead**

exception aiozk.exc.**Unimplemented**

exception aiozk.exc.**UnknownError**(*error_code*)

exception aiozk.exc.**UnknownSession**

exception aiozk.exc.**ZKError**

exception aiozk.exc.**ZKSystemError**

4.2 Recipe API References

4.2.1 DataWatcher

Todo

4.2.2 ChildrenWatcher

Todo

4.2.3 Lock

Todo

4.2.4 SharedLock

Todo

4.2.5 Lease

Todo

4.2.6 Barrier

Todo

4.2.7 DoubleBarrier

Todo

4.2.8 LeaderElection

Todo

4.2.9 Party

Todo

4.2.10 Counter

Todo

4.2.11 TreeCache

Todo

4.2.12 Allocator

Todo

4.3 Contributing

4.3.1 Test suite

Todo

4.3.2 Documentation

You can edit documents by modifying reStructuredText markup under `docs` directory. Before sending a Pull Request about documentation you can preview your modification as follows:

```
$ cd docs
$ make html
```

And then you can see the output with your browser. Open file `file://...path/to/aiozk/docs/_build/html/index.html`

4.3.3 Pull Request

Follow steps below to send a pull request to aiozk repository.

1. Fork aiozk repository GitHub
2. Make a change
3. Run test code and check that every test passes
4. Commit your modification *(it would be better with clear and concise commit message)*
5. Push your local ref to your own repository forked from aiozk.
6. Make a Pull Request

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

a

aiozk.exc, 16

INDEX

A

ACL (*class in aiozk*), 14
aiozk.exc
    module, 16
APIError, 16
AuthFailed, 16

B

BadArguments, 16
BadVersion, 16
begin_transaction() (*aiozk.ZKClient method*), 12

C

check_version() (*aiozk.transaction.Transaction method*), 12
close() (*aiozk.ZKClient method*), 9
commit() (*aiozk.transaction.Transaction method*), 12
ConnectError, 16
ConnectionLoss, 16
create() (*aiozk.transaction.Transaction method*), 13
create() (*aiozk.ZKClient method*), 10

D

DataInconsistency, 16
delete() (*aiozk.transaction.Transaction method*), 13
delete() (*aiozk.ZKClient method*), 10
deleteall() (*aiozk.ZKClient method*), 10

E

ensure_path() (*aiozk.ZKClient method*), 10
EphemeralOnLocalSession, 16
exists() (*aiozk.ZKClient method*), 9
exponential_backoff() (*aiozk.RetryPolicy class method*), 15

F

FailedRetry, 16
forever() (*aiozk.RetryPolicy class method*), 15

G

get() (*aiozk.ZKClient method*), 10
get_acl() (*aiozk.ZKClient method*), 12
get_children() (*aiozk.ZKClient method*), 11
get_data() (*aiozk.ZKClient method*), 11

I

InvalidACL, 16
InvalidCallback, 16
InvalidClientState, 16
InvalidStateTransition, 16

M

make() (*aiozk.ACL class method*), 14
MarshallingError, 16
module
    aiozk.exc, 16

N

n_times() (*aiozk.RetryPolicy class method*), 15
NewConfigNoQuorum, 16
NoAuth, 16
NoChildrenForEphemerals, 16
NodeExists, 16
NoNode, 16
NoServersError, 16
NotEmpty, 16
NotReadOnly, 16
NoWatcher, 16

O

once() (*aiozk.RetryPolicy class method*), 15
OperationTimeout, 16

R

ReconfigInProcess, 16
ResponseError, 16
Result (*class in aiozk.transaction*), 13
RetryPolicy (*class in aiozk*), 15
RolledBack, 16
RuntimeInconsistency, 16

S

SessionExpired, 16

SessionLost, 16
SessionMoved, 16
set() (*aiozk.ZKClient method*), 11
set_acl() (*aiozk.ZKClient method*), 12
set_data() (*aiozk.transaction.Transaction method*), 13
set_data() (*aiozk.ZKClient method*), 11
start() (*aiozk.ZKClient method*), 9
Stat (*class in aiozk.protocol.stat*), 14

T

TimeoutError, 16
Transaction (*class in aiozk.transaction*), 12
TransactionFailed, 16

U

UnfinishedRead, 16
Unimplemented, 16
UnknownError, 16
UnknownSession, 17
until_elapsed() (*aiozk.RetryPolicy class method*), 15

Z

ZKClient (*class in aiozk*), 9
ZKError, 17
ZKSystemError, 17