
k2hash

Release 1.0.0

Hiroataka Wakabayashi, Takeshi Nakatani

Mar 03, 2022

CONTENTS:

1	k2hash_python	1
1.1	Overview	1
1.2	Install	2
1.3	Usage	2
1.4	Development	2
1.5	Documents	2
1.6	Packages	2
1.7	License	3
1.8	AntPickax	3
2	k2hash	5
2.1	k2hash package	5
3	Credits	13
3.1	Development Lead	13
3.2	Contributors	13
4	History	15
4.1	1.0.0 (2022-02-07)	15
5	Indices and tables	17
	Python Module Index	19
	Index	21

K2HASH_PYTHON

1.1 Overview

k2hash_python is an official python driver for [k2hash](#).



- 🗄️ **Memory or File**
- ⚙️ **SubKey**
Binary Data
Archive
Transaction
Encrypt
Queue
- ⚙️ **Multi-Process**
Multi-Thread

1.2 Install

Firstly you must install the k2hash shared library:

```
curl -o- https://raw.githubusercontent.com/yahoojapan/k2hash_python/master/utils/libk2hash.sh | bash
```

Then, let's install k2hash using pip:

```
pip install k2hash
```

1.3 Usage

Try to set a key and get it:

```
import k2hash

k = k2hash.K2hash('test.k2h')
k.set('hello', 'world')
v = k.get('hello')
print(v)    // world
```

1.4 Development

Clone this repository and go into the directory, then run the following command:

```
$ python3 -m pip install --upgrade build
$ python3 -m build
```

1.5 Documents

Here are documents including other components.

[Document top page](#)

[About K2HASH](#)

[About AntPickax](#)

1.6 Packages

Here are packages including other components.

[k2hash\(python packages\)](#)

1.7 License

MIT License. See the LICENSE file.

1.8 AntPickax

k2hash_python is a project by [AntPickax](#), which is an open source team in [Yahoo Japan Corporation](#).

K2HASH

2.1 k2hash package

2.1.1 Submodules

2.1.2 k2hash.k2hash module

K2hash Python Driver under MIT License

```
class k2hash.k2hash.K2hash(k2hfile="", flag=None, readonly=True, removefile=True, fullmap=True,  
                           maskbit=8, cmaskbit=4, maxelementcnt=1024, pagesize=512, waitms=0,  
                           logfile="")
```

Bases: object

K2hash class provides methods to handle key/value pairs in k2hash hash database.

K2H_INVALID_HANDLE = 0

add_attribute_plugin_lib(*path*)
Adds a shared library that handles an attribute

add_decryption_password(*password*)
Adds a passphrase to decrypt a value.

add_subkey(*key, subkey, subval, password=None, expire_duration=None, time_unit=TimeUnit.SECONDS*)
Adds subkeys to a key/value pair.

begin_tx(*txfile, prefix=None, param=None, expire_duration=None*)
Starts a transaction logging.

close()
Closes a k2h file.

static create(*pathname, maskbit=8, cmaskbit=4, maxelementcnt=1024, pagesize=512*)
Creates a k2hash file.

dump_to_file(*path, is_skip_error=True*)
Dumps data to a file.

enable_encryption(*enable=True*)
Enables a feature to encrypt a value.

enable_history(*enable=True*)
Enables a feature to record a key modification history.

enable_mtime(*enable=True*)
Enables a feature to record value modification time.

get(*key*, *password=None*)
Gets the value

get_attributes(*key*, *use_str=True*)
Gets attributes of a key.

get_iterator(*key=None*)
Returns the k2hash iterator

get_subkeys(*key*, *use_str=True*)
Gets keys of subkeys of a key.

get_tx_file_fd()
Gets a transaction log file descriptor.

static get_tx_pool_size()
Gets the number of transaction thread pool.

property handle
Returns a Queue handle.

property libc
returns libc handle

property libk2hash
returns libk2hash handle

load_from_file(*path*, *is_skip_error=True*)
Loads data from a file.

print_attribute_plugins()
Prints attribute plugins to stderr.

print_attributes()
Prints attributes to stderr.

print_data_stats()
Prints data statistics.

print_table_stats(*level=DumpLevel.HEADER*)
Prints k2hash key table information.

remove(*key*, *remove_all_subkeys=False*)
Removes a key.

remove_subkeys(*key*, *subkeys*)
Removes subkeys from the key.

rename(*key*, *newkey*)
Renames a key with a new key.

set(*key*, *val*, *password=None*, *expire_duration=None*, *time_unit=TimeUnit.SECONDS*)
Sets a key/value pair

set_attribute(*key*, *attr_name*, *attr_val*)
Sets an attribute of a key.

set_default_encryption_password(*password*)
Sets the default encryption passphrase.

set_encryption_password_file(*path*)
Sets the data encryption password file.

set_expiration_duration(*expire_duration, time_unit=TimeUnit.SECONDS*)

Sets the duration to expire a value.

set_log_level(*level=LogLevel.INFO*)

Creates a k2hash file.

set_subkeys(*key, subkeys, password=None, expire_duration=None, time_unit=TimeUnit.SECONDS*)

Sets subkeys.

static set_tx_pool_size(*size*)

Sets the number of transaction thread pool.

stop_tx()

Stops a transaction logging.

static version()

Prints version information.

class k2hash.k2hash.**K2hashIterator**(*k2h, key=None*)

Bases: object

implements iterator of k2hash

2.1.3 k2hash.keyqueue module

K2hash Python Driver under MIT License

class k2hash.keyqueue.**KeyQueue**(*k2h, fifo=True, prefix=None, password=None, expire_duration=None*)

Bases: object

KeyQueue class provides methods to handle key/value pairs in k2hash hash database.

clear()

Removes all of the elements from this collection (optional operation).

close()

Free QueueHandle

element(*position=0*)

Finds and gets a object from the head of this queue.

empty()

Returns true if, and only if, queue size is 0.

get()

Finds and gets a object from the head of this queue.

property handle

Returns a Queue handle.

print()

Print the objects in this queue.

put(*obj*)

Inserts an element into the tail of this queue.

qsize()

Returns the number of queue.

remove(*count=1*)

Removes objects from this queue.

2.1.4 k2hash.queue module

K2hash Python Driver under MIT License

class k2hash.queue.**Queue**(*k2h, fifo=True, prefix=None, password=None, expire_duration=None*)

Bases: object

Queue class provides methods to handle key/value pairs in k2hash hash database.

clear()

Removes all of the elements from this collection (optional operation).

close()

Free QueueHandle

element(*position=0*)

Finds and gets a object from the head of this queue.

empty()

Returns true if, and only if, queue size is 0.

get()

Finds and gets a object from the head of this queue.

property handle

Returns a Queue handle.

print()

Print the objects in this queue.

put(*obj, attrs=None*)

Inserts an element into the tail of this queue.

qsize()

Returns the number of queue.

remove(*count=1*)

Removes objects from this queue.

2.1.5 Module contents

k2hash package

class k2hash.**AttrPack**

Bases: `_ctypes.Structure`

C Attr structure

keylength

Structure/Union member

pkey

Structure/Union member

pval

Structure/Union member

vallength

Structure/Union member

class k2hash.**DumpLevel**(*value*)

Bases: `enum.Enum`

k2hash file status information

ELEMENT = 4

HASH_TABLE = 2

HEADER = 1

PAGE = 5

SUB_HASH_TABLE = 3

```
class k2hash.K2hash(k2hfile="", flag=None, readonly=True, removefile=True, fullmap=True, maskbit=8,
                  cmaskbit=4, maxelementcnt=1024, pagesize=512, waitms=0, logfile="")
```

Bases: object

K2hash class provides methods to handle key/value pairs in k2hash hash database.

K2H_INVALID_HANDLE = 0

add_attribute_plugin_lib(*path*)

Adds a shared library that handles an attribute

add_decryption_password(*password*)

Adds a passphrase to decrypt a value.

add_subkey(*key*, *subkey*, *subval*, *password=None*, *expire_duration=None*, *time_unit=TimeUnit.SECONDS*)

Adds subkeys to a key/value pair.

begin_tx(*txfile*, *prefix=None*, *param=None*, *expire_duration=None*)

Starts a transaction logging.

close()

Closes a k2h file.

static create(*pathname*, *maskbit=8*, *cmaskbit=4*, *maxelementcnt=1024*, *pagesize=512*)

Creates a k2hash file.

dump_to_file(*path*, *is_skip_error=True*)

Dumps data to a file.

enable_encryption(*enable=True*)

Enables a feature to encrypt a value.

enable_history(*enable=True*)

Enables a feature to record a key modification history.

enable_mtime(*enable=True*)

Enables a feature to record value modification time.

get(*key*, *password=None*)

Gets the value

get_attributes(*key*, *use_str=True*)

Gets attributes of a key.

get_iterator(*key=None*)

Returns the k2hash iterator

get_subkeys(*key*, *use_str=True*)

Gets keys of subkeys of a key.

get_tx_file_fd()

Gets a transaction log file descriptor.

static get_tx_pool_size()
Gets the number of transaction thread pool.

property handle
Returns a Queue handle.

property libc
returns libc handle

property libk2hash
returns libk2hash handle

load_from_file(path, is_skip_error=True)
Loads data from a file.

print_attribute_plugins()
Prints attribute plugins to stderr.

print_attributes()
Prints attributes to stderr.

print_data_stats()
Prints data statistics.

print_table_stats(level=DumpLevel.HEADER)
Prints k2hash key table information.

remove(key, remove_all_subkeys=False)
Removes a key.

remove_subkeys(key, subkeys)
Removes subkeys from the key.

rename(key, newkey)
Renames a key with a new key.

set(key, val, password=None, expire_duration=None, time_unit=TimeUnit.SECONDS)
Sets a key/value pair

set_attribute(key, attr_name, attr_val)
Sets an attribute of a key.

set_default_encryption_password(password)
Sets the default encryption passphrase.

set_encryption_password_file(path)
Sets the data encryption password file.

set_expiration_duration(expire_duration, time_unit=TimeUnit.SECONDS)
Sets the duration to expire a value.

set_log_level(level=LogLevel.INFO)
Creates a k2hash file.

set_subkeys(key, subkeys, password=None, expire_duration=None, time_unit=TimeUnit.SECONDS)
Sets subkeys.

static set_tx_pool_size(size)
Sets the number of transaction thread pool.

stop_tx()
Stops a transaction logging.

```

    static version()
        Prints version information.
class k2hash.K2hashIterator(k2h, key=None)
    Bases: object
    implements iterator of k2hash
class k2hash.KeyPack
    Bases: ctypes.Structure
    C KeyPack structure
    length
        Structure/Union member
    pkey
        Structure/Union member
class k2hash.KeyQueue(k2h, fifo=True, prefix=None, password=None, expire_duration=None)
    Bases: object
    KeyQueue class provides methods to handle key/value pairs in k2hash hash database.
    clear()
        Removes all of the elements from this collection (optional operation).
    close()
        Free QueueHandle
    element(position=0)
        Finds and gets a object from the head of this queue.
    empty()
        Returns true if, and only if, queue size is 0.
    get()
        Finds and gets a object from the head of this queue.
    property handle
        Returns a Queue handle.
    print()
        Print the objects in this queue.
    put(obj)
        Inserts an element into the tail of this queue.
    qsize()
        Returns the number of queue.
    remove(count=1)
        Removes objects from this queue.
class k2hash.LogLevel(value)
    Bases: enum.Enum
    k2hash log level
    DEBUG = 5
    ERROR = 2
    INFO = 4
    SILENT = 1

```

WARNING = 3

class k2hash.OpenFlag(*value*)

Bases: `enum.Enum`

k2hash file open flags

EDIT = 2

MEMORY = 4

READ = 1

TEMPFILE = 3

class k2hash.Queue(*k2h, fifo=True, prefix=None, password=None, expire_duration=None*)

Bases: `object`

Queue class provides methods to handle key/value pairs in k2hash hash database.

clear()

Removes all of the elements from this collection (optional operation).

close()

Free QueueHandle

element(*position=0*)

Finds and gets a object from the head of this queue.

empty()

Returns true if, and only if, queue size is 0.

get()

Finds and gets a object from the head of this queue.

property handle

Returns a Queue handle.

print()

Print the objects in this queue.

put(*obj, attrs=None*)

Inserts an element into the tail of this queue.

qsize()

Returns the number of queue.

remove(*count=1*)

Removes objects from this queue.

class k2hash.TimeUnit(*value*)

Bases: `enum.Enum`

k2hash time units

DAYS = 1

HOURS = 2

MILLISECONDS = 3

MINUTES = 4

SECONDS = 5

CREDITS

3.1 Development Lead

- Hirotaka Wakabayashi <hiwakaba@yahoo-corp.jp>

3.2 Contributors

- Takeshi Nakatani <ggtakec@gmail.com>

HISTORY

4.1 1.0.0 (2022-02-07)

- First release on PyPI.

INDICES AND TABLES

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

PYTHON MODULE INDEX

k

`k2hash`, 8

`k2hash.k2hash`, 5

`k2hash.keyqueue`, 7

`k2hash.queue`, 8

A

add_attribute_plugin_lib() (k2hash.K2hash method), 9
 add_attribute_plugin_lib() (k2hash.k2hash.K2hash method), 5
 add_decryption_password() (k2hash.K2hash method), 9
 add_decryption_password() (k2hash.k2hash.K2hash method), 5
 add_subkey() (k2hash.K2hash method), 9
 add_subkey() (k2hash.k2hash.K2hash method), 5
 AttrPack (class in k2hash), 8

B

begin_tx() (k2hash.K2hash method), 9
 begin_tx() (k2hash.k2hash.K2hash method), 5

C

clear() (k2hash.KeyQueue method), 11
 clear() (k2hash.keyqueue.KeyQueue method), 7
 clear() (k2hash.Queue method), 12
 clear() (k2hash.queue.Queue method), 8
 close() (k2hash.K2hash method), 9
 close() (k2hash.k2hash.K2hash method), 5
 close() (k2hash.KeyQueue method), 11
 close() (k2hash.keyqueue.KeyQueue method), 7
 close() (k2hash.Queue method), 12
 close() (k2hash.queue.Queue method), 8
 create() (k2hash.K2hash static method), 9
 create() (k2hash.k2hash.K2hash static method), 5

D

DAYS (k2hash.TimeUnit attribute), 12
 DEBUG (k2hash.LogLevel attribute), 11
 dump_to_file() (k2hash.K2hash method), 9
 dump_to_file() (k2hash.k2hash.K2hash method), 5
 DumpLevel (class in k2hash), 8

E

EDIT (k2hash.OpenFlag attribute), 12
 ELEMENT (k2hash.DumpLevel attribute), 9

element() (k2hash.KeyQueue method), 11
 element() (k2hash.keyqueue.KeyQueue method), 7
 element() (k2hash.Queue method), 12
 element() (k2hash.queue.Queue method), 8
 empty() (k2hash.KeyQueue method), 11
 empty() (k2hash.keyqueue.KeyQueue method), 7
 empty() (k2hash.Queue method), 12
 empty() (k2hash.queue.Queue method), 8
 enable_encryption() (k2hash.K2hash method), 9
 enable_encryption() (k2hash.k2hash.K2hash method), 5
 enable_history() (k2hash.K2hash method), 9
 enable_history() (k2hash.k2hash.K2hash method), 5
 enable_mtime() (k2hash.K2hash method), 9
 enable_mtime() (k2hash.k2hash.K2hash method), 5
 ERROR (k2hash.LogLevel attribute), 11

G

get() (k2hash.K2hash method), 9
 get() (k2hash.k2hash.K2hash method), 5
 get() (k2hash.KeyQueue method), 11
 get() (k2hash.keyqueue.KeyQueue method), 7
 get() (k2hash.Queue method), 12
 get() (k2hash.queue.Queue method), 8
 get_attributes() (k2hash.K2hash method), 9
 get_attributes() (k2hash.k2hash.K2hash method), 6
 get_iterator() (k2hash.K2hash method), 9
 get_iterator() (k2hash.k2hash.K2hash method), 6
 get_subkeys() (k2hash.K2hash method), 9
 get_subkeys() (k2hash.k2hash.K2hash method), 6
 get_tx_file_fd() (k2hash.K2hash method), 9
 get_tx_file_fd() (k2hash.k2hash.K2hash method), 6
 get_tx_pool_size() (k2hash.K2hash static method), 9
 get_tx_pool_size() (k2hash.k2hash.K2hash static method), 6

H

handle (k2hash.K2hash property), 10
 handle (k2hash.k2hash.K2hash property), 6
 handle (k2hash.KeyQueue property), 11
 handle (k2hash.keyqueue.KeyQueue property), 7
 handle (k2hash.Queue property), 12

handle (*k2hash.queue.Queue* property), 8
HASH_TABLE (*k2hash.DumpLevel* attribute), 9
HEADER (*k2hash.DumpLevel* attribute), 9
HOURS (*k2hash.TimeUnit* attribute), 12

I

INFO (*k2hash.LogLevel* attribute), 11

K

K2H_INVALID_HANDLE (*k2hash.K2hash* attribute), 9
K2H_INVALID_HANDLE (*k2hash.k2hash.K2hash* attribute), 5
k2hash
 module, 8
K2hash (*class* in *k2hash*), 9
K2hash (*class* in *k2hash.k2hash*), 5
k2hash.k2hash
 module, 5
k2hash.keyqueue
 module, 7
k2hash.queue
 module, 8
K2hashIterator (*class* in *k2hash*), 11
K2hashIterator (*class* in *k2hash.k2hash*), 7
keylength (*k2hash.AttrPack* attribute), 8
KeyPack (*class* in *k2hash*), 11
KeyQueue (*class* in *k2hash*), 11
KeyQueue (*class* in *k2hash.keyqueue*), 7

L

length (*k2hash.KeyPack* attribute), 11
libc (*k2hash.K2hash* property), 10
libc (*k2hash.k2hash.K2hash* property), 6
libk2hash (*k2hash.K2hash* property), 10
libk2hash (*k2hash.k2hash.K2hash* property), 6
load_from_file() (*k2hash.K2hash* method), 10
load_from_file() (*k2hash.k2hash.K2hash* method), 6
LogLevel (*class* in *k2hash*), 11

M

MEMORY (*k2hash.OpenFlag* attribute), 12
MILLISECONDS (*k2hash.TimeUnit* attribute), 12
MINUTES (*k2hash.TimeUnit* attribute), 12
module
 k2hash, 8
 k2hash.k2hash, 5
 k2hash.keyqueue, 7
 k2hash.queue, 8

O

OpenFlag (*class* in *k2hash*), 12

P

PAGE (*k2hash.DumpLevel* attribute), 9

pkey (*k2hash.AttrPack* attribute), 8
pkey (*k2hash.KeyPack* attribute), 11
print() (*k2hash.KeyQueue* method), 11
print() (*k2hash.keyqueue.KeyQueue* method), 7
print() (*k2hash.Queue* method), 12
print() (*k2hash.queue.Queue* method), 8
print_attribute_plugins() (*k2hash.K2hash* method), 10
print_attribute_plugins() (*k2hash.k2hash.K2hash* method), 6
print_attributes() (*k2hash.K2hash* method), 10
print_attributes() (*k2hash.k2hash.K2hash* method), 6
print_data_stats() (*k2hash.K2hash* method), 10
print_data_stats() (*k2hash.k2hash.K2hash* method), 6
print_table_stats() (*k2hash.K2hash* method), 10
print_table_stats() (*k2hash.k2hash.K2hash* method), 6
put() (*k2hash.KeyQueue* method), 11
put() (*k2hash.keyqueue.KeyQueue* method), 7
put() (*k2hash.Queue* method), 12
put() (*k2hash.queue.Queue* method), 8
pval (*k2hash.AttrPack* attribute), 8

Q

qsize() (*k2hash.KeyQueue* method), 11
qsize() (*k2hash.keyqueue.KeyQueue* method), 7
qsize() (*k2hash.Queue* method), 12
qsize() (*k2hash.queue.Queue* method), 8
Queue (*class* in *k2hash*), 12
Queue (*class* in *k2hash.queue*), 8

R

READ (*k2hash.OpenFlag* attribute), 12
remove() (*k2hash.K2hash* method), 10
remove() (*k2hash.k2hash.K2hash* method), 6
remove() (*k2hash.KeyQueue* method), 11
remove() (*k2hash.keyqueue.KeyQueue* method), 7
remove() (*k2hash.Queue* method), 12
remove() (*k2hash.queue.Queue* method), 8
remove_subkeys() (*k2hash.K2hash* method), 10
remove_subkeys() (*k2hash.k2hash.K2hash* method), 6
rename() (*k2hash.K2hash* method), 10
rename() (*k2hash.k2hash.K2hash* method), 6

S

SECONDS (*k2hash.TimeUnit* attribute), 12
set() (*k2hash.K2hash* method), 10
set() (*k2hash.k2hash.K2hash* method), 6
set_attribute() (*k2hash.K2hash* method), 10
set_attribute() (*k2hash.k2hash.K2hash* method), 6
set_default_encryption_password() (*k2hash.K2hash* method), 10

set_default_encryption_password()
 (*k2hash.k2hash.K2hash method*), 6
 set_encryption_password_file() (*k2hash.K2hash
 method*), 10
 set_encryption_password_file()
 (*k2hash.k2hash.K2hash method*), 6
 set_expiration_duration() (*k2hash.K2hash
 method*), 10
 set_expiration_duration() (*k2hash.k2hash.K2hash
 method*), 6
 set_log_level() (*k2hash.K2hash method*), 10
 set_log_level() (*k2hash.k2hash.K2hash method*), 7
 set_subkeys() (*k2hash.K2hash method*), 10
 set_subkeys() (*k2hash.k2hash.K2hash method*), 7
 set_tx_pool_size() (*k2hash.K2hash static method*),
 10
 set_tx_pool_size() (*k2hash.k2hash.K2hash static
 method*), 7
 SILENT (*k2hash.LogLevel attribute*), 11
 stop_tx() (*k2hash.K2hash method*), 10
 stop_tx() (*k2hash.k2hash.K2hash method*), 7
 SUB_HASH_TABLE (*k2hash.DumpLevel attribute*), 9

T

TEMPFILE (*k2hash.OpenFlag attribute*), 12
 TimeUnit (*class in k2hash*), 12

V

vallength (*k2hash.AttrPack attribute*), 8
 version() (*k2hash.K2hash static method*), 10
 version() (*k2hash.k2hash.K2hash static method*), 7

W

WARNING (*k2hash.LogLevel attribute*), 11