YEDB Documentation

Release 0.2.25

AlterTech

Sep 30, 2022

CONTENTS

exception yedb.**ChecksumError**

exception yedb.**FieldNotFound**

class yedb.**KeyDict**(*db*, *key*)

Dictionary key object

Should not be used directly, better usage:

with db.key_as_dict('path.to.key') as key:
do something

Direct acccess to key dictionary is possible via obj.data. If any fields are modified during the direct access, calling obj.set_modified() is REQUIRED (otherwise the data will be not written back when the object is closed)

delete(*name*)

Delete key field

Doesn't raise any exceptions if the field is not present

get(*name*, *default=<class 'KeyError'>*)

Get key field

Parameters

- **name** – field name
- **default** – default value, if the field is not present (if not specified, KeyError is raised)

set(*name*, *value*)

Set key field

Parameters

- **name** – field name
- **value** – field value

class yedb.**KeyList**(*db*, *key*)

List key object

Should not be used directly, better usage:

with db.key_as_list('path.to.key') as key:
do something

Direct acccess to key list is possible via obj.data. If the data is modified during the direct access, calling obj.set_modified() is REQUIRED (otherwise the data will be not written back when the object is closed)

append(*value*)

Append value to list

remove(*value*)

Remove value from list

exception yedb.**SchemaValidationError**

class yedb.**Session**(*db*)

Session object, all methods except open/close are proxied to db

close()

Close session

open()

Open session

class yedb.**YEDB**(*path*, *default_fmt='json'*, *default_checksums=True*, ***kwargs*)

File-based database

The object is thread-safe

Create / open database

Data formats supported:

json: JSON (uses rapidjson module if present), default yaml, yml: YAML (requires "pyyaml" module) msgpack: MessagePack (requires "msgpack-python" module) cbor: CBOR (requires "cbor" module) pickle: Python's native pickle

Can be used either directly or via with statement:

with yedb.YEDB('/path/to/db1') as db:
do something

Key parts are split with "/" symbols

If path is specified as HTTP/HTTPS URI, the object transforms itself into JSON RPC client (methods, not listed at yedb.common.METHODS become unimplemented)

Parameters

- **path** – database directory
- **lock_path** – lock file path (default: path / db.lock)
- **default_fmt** – default data format
- **default_checksums** – use SHA256 checksums by default
- **timeout** – server timeout (for client/server mode)
- **http_username** – http username
- **http_password** – http password
- **http_auth** – auth type (basic or digest)
- **cache_size** – item cache size

__enter__(**args*, ***kwargs*)

Raises
TimeoutError –

check()

Check database

Returns
Generator object with broken keys found

convert_fmt(*new_fmt*, *checksums=True*)

Convert database format

Parameters

- **new_fmt** – new format
- **checksums** – use checksums (default: True)

Returns
Generator object with tuples (key, True|False) where True means a key is converted and False means a key (old-format) is purged.

`do_repair()`

One-shot auto repair

Calls repair and logs the details

Returns
True if repair is successful, False if an error occured. Does not raise exceptions, as the broken database is still usable, except may miss some keys or they may be broken.

`key_as_dict(key)`

Returns KeyDict object

Note: doesn't lock the key on client/server

Parameters
key – key name

`key_as_list(key)`

Returns KeyList object

Note: doesn't lock the key on client/server

Parameters
key – key name

`key_copy(key, dst_key)`

Copy key to new

`key_delete(key)`

Deletes key

Parameters
key – key name

`key_delete_field(key, field)`

Delete key field value

The key file is always overriden

Parameters

- **key** – key name
- **field** – field name
- **value** – key value

`key_delete_recursive(key)`

Deletes key and its subkeys

Parameters
key – key name

`key_dump(key='')`

Equal to get_subkeys(ignore_broken=True, hidden=False)

key_exists(*key*)

Returns
if key exists False: if not

Return type
True

key_explain(*key*)

Get key value + extended info

Parameters
name – key name

Returns
dict(value, info=Path.stat, checksum=checksum, file=Path)

key_get(*key*, *default=<class 'KeyError'>*)

Get key value

Parameters

- **key** – key name
- **default** – default value, if the field is not present (if not specified, KeyError is raised)

key_get_field(*key*, *field*, *default=<class 'KeyError'>*)

Get key field value

Parameters

- **key** – key name
- **field** – key field name
- **default** – default value, if the field is not present (if not specified, KeyError is raised)

key_get_recursive(*key=''*, *_ignore_broken=False*)

Get subkeys of the specified key and their values (including the key itself)

Parameters
key – key name, if not specified, all keys / values are returned

Returns

A generator object is returned, so the db becomes locked until all values are yielded. To unlock the db earlier, convert the returned generator into a list

Generated values are returned as tuples (key_name, key_value)

key_list(*key=''*)

List subkeys of the specified key (including the key itself)

Parameters
key – key name, if not specified, all keys are returned

Returns
A generator object is returned, so the db becomes locked until all values are yielded. To unlock the db earlier, convert the returned generator into a list

key_list_all(*key=''*)

List subkeys of the specified key (including the key itself), including hidden

key_load(*data*)

Loads keys

Schema validations are ignored

Parameters

data – list or generator of key/value pairs (lists or tuples)

key_rename(*key*, *dst_key*)

Rename key or category to new

key_set(*key*, *value*, *_stime=None*, *_ignore_schema=False*)

Set key value

The key file is always overriden

Parameters

- **key** – key name
- **value** – key value

key_set_field(*key*, *field*, *value*)

Set key field value

The key file is always overriden

Parameters

- **key** – key name
- **field** – field name
- **value** – key value

key_update(*key*, *data*)

Updates dict key with values in data

Parameters

data – dict

open(*auto_create=True*, *auto_repair=False*, *_skip_lock=False*, *_force_lock_ex=False*, *_skip_meta=False*, ***kwargs*)

Parameters

- **auto_create** – automatically create db
- **auto_repair** – automatically repair db
- **auto_flush** – always flush written data to disk
- **lock_ex** – lock database exclusively, so no other thread/process can open it (requires "portalocker" module)

Raises

- **TimeoutError** – database lock timeout
- **ModuleNotFoundError** – missing Python module for the chosen format
- **ValueError** – Unsupported format chosen
- **RuntimeError** – database / meta info errors

purge(*_keep_broken=False*)

Purges empty directories

When keys are deleted, unnecessary directories are usually auto-purged, but in case of errors this method can be called to manually purge empty dirs

Also deletes unnecessary files (e.g. left after format conversion) and checks all entries.

The command also clears memory cache.

Returns
Generator object with broken keys found and removed

purge_cache()

Purge cache only

repair()

Repairs database

Finds temp key files and tries to repair them if they are valid. Requires checksums enabled

Returns
Generator object with tuples (key, True|False) where True means a key is repaired and False means a key is purged.

safe_purge()

Same as purge, but keeps broken keys

session()

Get session object

PYTHON MODULE INDEX

y

INDEX