# aiocouchdb Documentation

***Release 0.8.0***

**Alexander Shorin**

March 21, 2015

Contents

**source** https://github.com/kxepal/aiocouchdb

**documentation** http://aiocouchdb.readthedocs.org/en/latest/

**license** BSD

CHAPTER 1

# CouchDB 1.x API

## 1.1 Server

**class** aiocouchdb.v1.server.**Server**(*url_or_resource='http://localhost:5984'*, ***, *authdb_class=None*, *authdb_name=None*, *config_class=None*, *database_class=None*, *session_class=None*)

Implementation of *CouchDB Server API*.

**active_tasks**(***, *auth=None*)

Returns list of *active tasks* which runs on server.

**Parameters** **auth** – aiocouchdb.authn.AuthProvider instance

**Return type** list

**all_dbs**(***, *auth=None*)

Returns list of available *databases* on server.

**Parameters** **auth** – aiocouchdb.authn.AuthProvider instance

**Return type** list

**authdb**

Proxy to the authentication database instance.

**authdb_class**

Authentication database class

alias of AuthDatabase

**authdb_name = '_users'**

Authentication database name

**config**

Proxy to the related config_class instance.

**config_class**

Default ServerConfig instance class

alias of ServerConfig

**database_class**

Default Database instance class

alias of Database

**db**(*dbname*, *, *auth=None*)
Returns `Database` instance against specified database name.

If database isn't accessible for provided auth credentials, this method raises `aiocouchdb.errors.HttpErrorException` with the related response status code.

**Parameters**

- **dbname** (*str*) – Database name
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** `aiocouchdb.v1.server.Server.database_class`

**db_updates**(*, *auth=None*, *feed_buffer_size=None*, *feed=None*, *timeout=None*, *heartbeat=None*)
Emits *databases events* for the related server instance.

**Parameters**

- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **feed** (*str*) – Feed type
- **timeout** (*int*) – Timeout in milliseconds
- **heartbeat** (*bool*) – Whenever use heartbeats to keep connection alive

Depending on feed type returns:

- `dict` - for default or `longpoll` feed
- `aiocouchdb.feeds.JsonFeed` - for `continuous` feed
- `aiocouchdb.feeds.EventSourceFeed` - for `eventsource` feed

**info**(*, *auth=None*)
Returns server *meta information and welcome message*.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**log**(*, *bytes=None*, *offset=None*, *auth=None*)
Returns a chunk of data from the tail of *CouchDB's log* file.

**Parameters**

- **bytes** (*int*) – Bytes to return
- **offset** (*int*) – Offset in bytes where the log tail should be started
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** str

**replicate**(*source*, *target*, *, *auth=None*, *authobj=None*, *cancel=None*, *continuous=None*, *create_target=None*, *doc_ids=None*, *filter=None*, *headers=None*, *proxy=None*, *query_params=None*, *since_seq=None*, *checkpoint_interval=None*, *connection_timeout=None*, *http_connections=None*, *retries_per_request=None*, *socket_options=None*, *use_checkpoints=None*, *worker_batch_size=None*, *worker_processes=None*)
*Runs a replication* from `source` to `target`.

**Parameters**

- **source** (*str*) – Source database name or URL

- **target** (*str*) – Target database name or URL
- **auth** – `aiocouchdb.authn.AuthProvider` instance (don't confuse with `authobj` which belongs to replication options)
- **authobj** (*dict*) – Authentication object for the target database
- **cancel** (*bool*) – Cancels active replication
- **continuous** (*bool*) – Runs continuous replication
- **create_target** (*bool*) – Creates target database if it not exists
- **doc_ids** (*list*) – List of specific document ids to replicate
- **filter** (*str*) – Filter function name
- **headers** (*dict*) – Custom replication request headers
- **proxy** (*str*) – Proxy server URL
- **query_params** (*dict*) – Custom query parameters for filter function
- **since_seq** – Start replication from specified sequence number
- **checkpoint_interval** (*int*) – Tweaks checkpoint_interval option
- **connection_timeout** (*int*) – Tweaks connection_timeout option
- **http_connections** (*int*) – Tweaks http_connections option
- **retries_per_request** (*int*) – Tweaks retries_per_request option
- **socket_options** (*str*) – Tweaks socket_options option
- **use_checkpoints** (*bool*) – Tweaks use_checkpoints option
- **worker_batch_size** (*int*) – Tweaks worker_batch_size option
- **worker_processes** (*int*) – Tweaks worker_processes option

**Return type** dict

**restart**(*, *auth=None*)
*Restarts* server instance.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**session**
Proxy to the related `session_class` instance.

**session_class**
Default `Session` instance class

alias of `Session`

**stats**(*metric=None*, *, *auth=None*, *flush=None*, *range=None*)
Returns *server statistics*.

**Parameters**

- **metric** (*str*) – Metrics name in format `group/name`. For instance, `httpd/requests`. If omitted, all metrics will be returned
- **flush** (*bool*) – If `True`, collects samples right for this request
- **range** (*int*) – Sampling range

- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**uuids**(*, *auth=None*, *count=None*)
Returns *UUIDs* generated on server.

**Parameters**

- **count** (*int*) – Amount of UUIDs to generate
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** list

### 1.1.1 Configuration

**class** `aiocouchdb.v1.config.`**ServerConfig**(*resource*)
Implements */_config/** API. Should be used via `server.config` property.

**delete**(*section*, *key*, *, *auth=None*)
Deletes specific *configuration option* and returns it value back.

**Parameters**

- **section** (*string*) – Configuration section name
- **key** (*string*) – Option name
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** str

**exists**(*section*, *key*, *, *auth=None*)
Checks if *configuration option* exists.

**Parameters**

- **section** (*str*) – Section name
- **key** (*str*) – Option name
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**get**(*section=None*, *key=None*, *, *auth=None*)
Returns *server configuration*. Depending on specified arguments returns:

•*Complete configuration* if `section` and `key` are `None`

•*Section options* if `section` was specified

•*Option value* if both `section` and `key` were specified

**Parameters**

- **section** (*str*) – Section name (*optional*)
- **key** (*str*) – Option name (*optional*)
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict or str

**update**(*section*, *key*, *value*, *, *auth=None*)
Updates specific *configuration option* value and returns the old one back.

**Parameters**

- **section** (*str*) – Configuration section name
- **key** (*str*) – Option name
- **value** (*str*) – New option value
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** str

### 1.1.2 Session

**class** `aiocouchdb.v1.session.`**Session**(*resource*)
Implements */_session* API. Should be used via `server.session` property.

**close**(*, *auth=None*)
Closes active cookie session. Uses for `aiocouchdb.authn.CookieAuthProvider`.

**cookie_auth_provider_class**
alias of `CookieAuthProvider`

**info**(*, *auth=None*)
Returns information about authenticated user. Usable for any `AuthProvider`.

**Return type** dict

**open**(*name*, *password*)
Opens session for cookie auth provider and returns the auth provider back for usage in further requests.

**Parameters**

- **name** (*str*) – Username
- **password** (*str*) – User's password

**Return type** `aiocouchdb.authn.CookieAuthProvider`

## 1.2 Database

**class** `aiocouchdb.v1.database.`**Database**(*url_or_resource*, *, *dbname=None*, *document_class=None*, *design_document_class=None*, *security_class=None*, *view_class=None*)
Implementation of *CouchDB Database API*.

**all_docs**(**keys*, *auth=None*, *feed_buffer_size=None*, *att_encoding_info=None*, *attachments=None*, *conflicts=None*, *descending=None*, *endkey=Ellipsis*, *endkey_docid=None*, *include_docs=None*, *inclusive_end=None*, *limit=None*, *skip=None*, *stale=None*, *startkey=Ellipsis*, *startkey_docid=None*, *update_seq=None*)
Iterates over *all documents view*.

**Parameters**

- **keys** (*str*) – List of document ids to fetch. This method is smart enough to use *GET* or *POST* request depending on amount of `keys`
- **auth** – `aiocouchdb.authn.AuthProvider` instance

- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs
- **attachments** (*bool*) – Includes attachments content into documents. **Warning**: use with caution!
- **conflicts** (*bool*) – Includes conflicts information into documents
- **descending** (*bool*) – Return rows in descending by key order
- **endkey** (*str*) – Stop fetching rows when the specified key is reached
- **endkey_docid** (*str*) – Stop fetching rows when the specified document ID is reached
- **include_docs** (*str*) – Include document body for each row
- **inclusive_end** (*bool*) – When `False`, doesn't includes `endkey` in returned rows
- **limit** (*int*) – Limits the number of the returned rows by the specified number
- **skip** (*int*) – Skips specified number of rows before starting to return the actual result
- **stale** (*str*) – Allow to fetch the rows from a stale view, without triggering index update. Supported values: `ok` and `update_after`
- **startkey** (*str*) – Return rows starting with the specified key
- **startkey_docid** (*str*) – Return rows starting with the specified document ID
- **update_seq** (*bool*) – Include an `update_seq` value into view results header

**Return type** `aiocouchdb.feeds.ViewFeed`

**bulk_docs**(*docs*, ***, *auth=None*, *all_or_nothing=None*, *new_edits=None*)

*Updates multiple documents* using a single request.

**Parameters**

- **docs** (*Iterable*) – Sequence of document objects (`dict`)
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **all_or_nothing** (*bool*) – Sets the database commit mode to use *all-or-nothing* semantics
- **new_edits** (*bool*) – If *False*, prevents the database from assigning them new revision for updated documents

**Return type** list

**changes**(**doc_ids*, *auth=None*, *feed_buffer_size=None*, *att_encoding_info=None*, *attachments=None*, *conflicts=None*, *descending=None*, *feed=None*, *filter=None*, *headers=None*, *heartbeat=None*, *include_docs=None*, *limit=None*, *params=None*, *since=None*, *style=None*, *timeout=None*, *view=None*)

Emits *database changes events*.

**Parameters**

- **doc_ids** (*str*) – Document IDs to filter for. This method is smart enough to use *GET* or *POST* request depending if any `doc_ids` were provided or not and automatically sets `filter` param to `_doc_ids` value.
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs

- **attachments** (*bool*) – Includes the Base64-encoded content of an attachments in the documents
- **conflicts** (*bool*) – Includes conflicts information in the documents
- **descending** (*bool*) – Return changes in descending order
- **feed** (*str*) – *Changes feed type*
- **filter** (*str*) – Filter function name
- **headers** (*dict*) – Custom request headers
- **heartbeat** (*int*) – Period in milliseconds after which an empty line will be sent from server as the result to keep connection alive
- **include_docs** (*bool*) – Includes the associated document for each emitted event
- **limit** (*int*) – Limits a number of returned events by the specified value
- **since** – Starts listening changes feed since given *update sequence* value
- **params** (*dict*) – Custom request query parameters
- **style** (*str*) – Changes feed output style: `all_docs`, `main_only`
- **timeout** (*int*) – Period in milliseconds to await for new changes before close the feed. Works for continuous feeds
- **view** (*str*) – View function name which would be used as filter. Implicitly sets `filter` param to `_view` value

**Return type** `aiocouchdb.feeds.ChangesFeed`

**compact**(*ddoc_name=None*, *, *auth=None*)

Initiates *database* or *view index* compaction.

**Parameters**

- **ddoc_name** (*str*) – Design document name. If specified initiates view index compaction instead of database
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**create**(*, *auth=None*)

Creates a database.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**ddoc**(*docid*, *, *auth=None*)

Returns `DesignDocument` instance against specified document ID. This ID may startswith with `_design/` prefix and if it's not prefix will be added automatically.

If document isn't accessible for auth provided credentials, this method raises `aiocouchdb.errors.HttpErrorException` with the related response status code.

**Parameters**

- **docid** (*str*) – Document ID
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** `aiocouchdb.v1.database.Database.design_document_class`

**delete**(*, *auth=None*)
Deletes a database.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**design_document_class**
alias of `DesignDocument`

**doc**(*docid=None*, *, *auth=None*, *idfun=<function uuid4 at 0x7f6c95f78620>*)
Returns `Document` instance against specified document ID.

If document ID wasn't specified, the `idfun` function will be used to generate it.

If document isn't accessible for auth provided credentials, this method raises `aiocouchdb.errors.HttpErrorException` with the related response status code.

**Parameters**

- **docid** (*str*) – Document ID
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **idfun** – Document ID generation function. Should return `str` or other object which could be translated into string

**Return type** `aiocouchdb.v1.database.Database.document_class`

**document_class**
alias of `Document`

**ensure_full_commit**(*, *auth=None*)
Ensures that all bits are *committed on disk*.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**exists**(*, *auth=None*)
Checks if database exists on server. Assumes success on receiving response with *200 OK* status.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**info**(*, *auth=None*)
Returns database information.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**missing_revs**(*id_revs*, *, *auth=None*)
Returns *document missed revisions* in the database by given document-revisions mapping.

**Parameters**

- **id_revs** (*dict*) – Mapping between document ID and list of his revisions to search for.
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**name**
Returns a database name specified in class constructor.

**purge**(*id_revs*, ***, *auth=None*)
*Permanently removes specified document revisions* from the database.

**Parameters**

- **id_revs** (*dict*) – Mapping between document ID and list of his revisions to purge
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**revs_diff**(*id_revs*, ***, *auth=None*)
Returns *document revisions difference* in the database by given document-revisions mapping.

**Parameters**

- **id_revs** (*dict*) – Mapping between document ID and list of his revisions to compare
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**revs_limit**(*count=None*, ***, *auth=None*)
Returns the *limit of database revisions* to store or updates it if `count` parameter was specified.

**Parameters**

- **count** (*int*) – Amount of revisions to store
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** int or dict

**security**
Proxy to the related `security_class` instance.

**security_class**
alias of `DatabaseSecurity`

**temp_view**(*map_fun*, *red_fun=None*, *language=None*, ***, *auth=None*, *feed_buffer_size=None*, *att_encoding_info=None*, *attachments=None*, *conflicts=None*, *descending=None*, *endkey=Ellipsis*, *endkey_docid=None*, *group=None*, *group_level=None*, *include_docs=None*, *inclusive_end=None*, *keys=Ellipsis*, *limit=None*, *reduce=None*, *skip=None*, *stale=None*, *startkey=Ellipsis*, *startkey_docid=None*, *update_seq=None*)
Executes *temporary view* and returns it results according specified parameters.

**Parameters**

- **map_fun** (*str*) – Map function source code
- **red_fun** (*str*) – Reduce function source code
- **language** (*str*) – Query server language to process the view
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs
- **attachments** (*bool*) – Includes attachments content into documents. **Warning**: use with caution!
- **conflicts** (*bool*) – Includes conflicts information into documents
- **descending** (*bool*) – Return rows in descending by key order
- **endkey** – Stop fetching rows when the specified key is reached

- **endkey_docid** (*str*) – Stop fetching rows when the specified document ID is reached
- **group** (*bool*) – Reduces the view result grouping by unique keys
- **group_level** (*int*) – Reduces the view result grouping the keys with defined level
- **include_docs** (*str*) – Include document body for each row
- **inclusive_end** (*bool*) – When `False`, doesn't includes `endkey` in returned rows
- **keys** (*list*) – List of view keys to fetch
- **limit** (*int*) – Limits the number of the returned rows by the specified number
- **reduce** (*bool*) – Defines is the reduce function needs to be applied or not
- **skip** (*int*) – Skips specified number of rows before starting to return the actual result
- **stale** (*str*) – Allow to fetch the rows from a stale view, without triggering index update. Supported values: `ok` and `update_after`
- **startkey** – Return rows starting with the specified key
- **startkey_docid** (*str*) – Return rows starting with the specified document ID
- **update_seq** (*bool*) – Include an `update_seq` value into view results header

**Return type** `aiocouchdb.feeds.ViewFeed`

**view_class**
alias of `View`

**view_cleanup**(*, *auth=None*)
*Removes outdated views* index files.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**class** `aiocouchdb.v1.authdb.`**AuthDatabase**(*url_or_resource*, *, *dbname=None*, *document_class=None*, *design_document_class=None*, *security_class=None*, *view_class=None*)
Represents system authentication database. Used via `aiocouchdb.v1.server.Server.authdb`.

**document_class**
alias of `UserDocument`

### 1.2.1 Security

**class** `aiocouchdb.v1.security.`**DatabaseSecurity**(*resource*)
Provides set of methods to work with *database security API*. Should be used via `database.security` property.

**get**(*, *auth=None*)
Returns database security object.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**update**(*, *auth=None*, *admins=None*, *members=None*, *merge=False*)
Updates database security object.

**Parameters**

- **auth** – `aiocouchdb.authn.AuthProvider` instance

- **admins** (*dict*) – Mapping of administrators users/roles
- **members** (*dict*) – Mapping of members users/roles
- **merge** (*bool*) – Merges admins/members mappings with existed ones when is `True`, otherwise replaces them with the given

**Return type** dict

**update_admins**(*, *auth=None*, *names=None*, *roles=None*, *merge=False*)

Helper for `update()` method to update only database administrators leaving members as is.

**Parameters**

- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **names** (*list*) – List of user names
- **roles** (*list*) – List of role names
- **merge** (*bool*) – Merges user/role lists with existed ones when is `True`, otherwise replaces them with the given

**Return type** dict

**update_members**(*, *auth=None*, *names=None*, *roles=None*, *merge=False*)

Helper for `update()` method to update only database members leaving administrators as is.

**Parameters**

- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **names** (*list*) – List of user names
- **roles** (*list*) – List of role names
- **merge** (*bool*) – Merges user/role lists with existed ones when is `True`, otherwise replaces them with the given

**Return type** dict

## 1.3 Document

**class** aiocouchdb.v1.document.**Document**(*url_or_resource*, *, *docid=None*, *attachment_class=None*)

Implementation of *CouchDB Document API*.

**att**(*attname*, *, *auth=None*)

Returns `Attachment` instance against specified attachment.

If attachment isn't accessible for auth provided credentials, this method raises `aiocouchdb.errors.HttpErrorException` with the related response status code.

**Parameters**

- **attname** (*str*) – Attachment name
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** `aiocouchdb.v1.document.Document.attachment_class`

**attachment_class**

alias of `Attachment`

**copy**(*newid*, *rev=None*, ***, *auth=None*)
Copies a document with the new ID within the same database.

**Parameters**

- **newid** (*str*) – New document ID
- **rev** (*str*) – New document ID revision. Used for copying over existed document
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**delete**(*rev*, ***, *auth=None*, *preserve_content=None*)
Deletes a document from server.

By default document will be deleted using *DELETE* HTTP method. On this request CouchDB removes all document fields, leaving only system `_id` and `_rev` and adding `"_deleted": true` one. When *preserve_content* set to `True`, document will be marked as deleted (by adding `"_deleted": true` field without removing existed ones) via *PUT* request. This feature costs two requests to fetch and update the document and also such documents consumes more space by oblivious reasons.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **preserve_content** (*bool*) – Whenever to preserve document content on deletion

**Return type** dict

**exists**(*rev=None*, ***, *auth=None*)
Checks if document exists in the database. Assumes success on receiving response with *200 OK* status.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**get**(*rev=None*, ***, *auth=None*, *att_encoding_info=None*, *attachments=None*, *atts_since=None*, *conflicts=None*, *deleted_conflicts=None*, *local_seq=None*, *meta=None*, *open_revs=None*, *revs=None*, *revs_info=None*)
Returns a document object.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs
- **attachments** (*bool*) – Includes the Base64-encoded content of an attachments in the documents
- **atts_since** (*list*) – Includes attachments that was added since the specified revisions
- **conflicts** (*bool*) – Includes conflicts information in the documents
- **deleted_conflicts** (*bool*) – Includes information about deleted conflicted revisions in the document
- **local_seq** (*bool*) – Includes local sequence number in the document

- **meta** (*bool*) – Includes meta information in the document
- **open_revs** (*list*) – Returns the specified leaf revisions
- **revs** (*bool*) – Includes information about all known revisions
- **revs_info** (*bool*) – Includes information about all known revisions and their status

**Return type** dict or list if *open_revs* specified

**get_open_revs**(**open_revs*, *auth=None*, *att_encoding_info=None*, *atts_since=None*, *local_seq=None*, *revs=None*)

Returns document open revisions with their attachments.

Unlike `get(open_revs=[...])`, this method works with `multipart/mixed` response returning multipart reader which is more optimized to handle large data sets with lesser memory footprint.

Note, that this method always returns attachments along with leaf revisions.

**Parameters**

- **open_revs** (*list*) – Leaf revisions to return. If omitted, all leaf revisions will be returned
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **att_encoding_info** (*bool*) – Includes encoding information in an attachments stubs
- **atts_since** (*list*) – Includes attachments that was added since the specified revisions
- **local_seq** (*bool*) – Includes local sequence number in each document
- **revs** (*bool*) – Includes information about all known revisions in each document

**Return type** `OpenRevsMultipartReader`

**get_with_atts**(*rev=None*, ***, *auth=None*, *att_encoding_info=None*, *atts_since=None*, *conflicts=None*, *deleted_conflicts=None*, *local_seq=None*, *meta=None*, *revs=None*, *revs_info=None*)

Returns document with attachments.

This method is more optimal than `get(attachments=true)` since it uses multipart API and doesn't requires to read all the attachments, extract then from JSON document and decode from base64.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs
- **atts_since** (*list*) – Includes attachments that was added since the specified revisions
- **conflicts** (*bool*) – Includes conflicts information in the documents
- **deleted_conflicts** (*bool*) – Includes information about deleted conflicted revisions in the document
- **local_seq** (*bool*) – Includes local sequence number in the document
- **meta** (*bool*) – Includes meta information in the document
- **revs** (*bool*) – Includes information about all known revisions
- **revs_info** (*bool*) – Includes information about all known revisions and their status

**Return type** `DocAttachmentsMultipartReader`

**id**
Returns a document id specified in class constructor.

**modified**(*rev*, *, *auth=None*)
Checks if document was modified in database since specified revision.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**rev**(*, *auth=None*)
Returns current document revision by using HEAD request.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** str

**update**(*doc*, *, *atts=None*, *auth=None*, *batch=None*, *new_edits=None*, *rev=None*)
Updates a document on server.

**Parameters**

- **doc** (*dict*) – Document object. Should implement `MutableMapping` interface
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **atts** (*dict*) – Attachments mapping where keys are represents attachment name and value is file-like object or bytes
- **batch** (*str*) – Updates in batch mode (asynchronously) This argument accepts only `"ok"` value.
- **new_edits** (*bool*) – Signs about new document edition. When `False` allows to create conflicts manually
- **rev** (*str*) – Document revision. Optional, since document `_rev` field is also respected

**Return type** dict

> **Warning:** Updating document with attachments is not able to use all the advantages of multipart request due to COUCHDB-2295 issue, so don't even try to update a document with several gigabytes attachments with this method. Put them one-by-one via `aiocouchdb.v1.attachment.Attachment.update()` method.

**class** `aiocouchdb.v1.authdb.`**UserDocument**(**args*, ***kwargs*)
Represents user document for the `authentication database`.

**name**
Returns username.

**register**(*password*, *, *auth=None*, ***additional_data*)
Helper method over `aiocouchdb.v1.document.Document.update()` to change a user password.

**Parameters**

- **password** (*str*) – User's password
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**update_password**(*password*, *, *auth=None*)
Helper method over `aiocouchdb.v1.document.Document.update()` to change a user password.

**Parameters**

- **password** (*str*) – New password
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**class** `aiocouchdb.v1.document.`**DocAttachmentsMultipartReader**(*headers*, *content*)
Special multipart reader optimized for requesting single document with attachments. Matches output with `OpenRevsMultipartReader`.

**next**()
Emits a tuple of document object (`dict`) and multipart reader of the followed attachments (if any).

**Return type** tuple

**class** `aiocouchdb.v1.document.`**OpenRevsMultipartReader**(*headers*, *content*)
Special multipart reader optimized for reading document's open revisions with attachments.

**multipart_reader_cls**
alias of `MultipartReader`

**next**()
Emits a tuple of document object (`dict`) and multipart reader of the followed attachments (if any).

**Return type** tuple

## 1.4 Design Document

**class** `aiocouchdb.v1.designdoc.`**DesignDocument**(*url_or_resource*, *, *docid=None*, *document_class=None*, *view_class=None*)
Implementation of *CouchDB Design Document API*.

**doc**
Returns `document_class` instance to operate with design document as with regular CouchDB document.

**Return type** `Document`

**document_class**
alias of `Document`

**id**
Returns a document id specified in class constructor.

**info**(*, *auth=None*)
*Returns view index information*.

**Parameters** **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**list**(*list_name*, *view_name=None*, **keys*, *auth=None*, *headers=None*, *data=None*, *params=None*, *format=None*, *att_encoding_info=None*, *attachments=None*, *conflicts=None*, *descending=None*, *endkey=Ellipsis*, *endkey_docid=None*, *group=None*, *group_level=None*, *include_docs=None*, *inclusive_end=None*, *limit=None*, *reduce=None*, *skip=None*, *stale=None*, *startkey=Ellipsis*, *startkey_docid=None*, *update_seq=None*)

Calls a *list function* and returns a raw response object.

**Parameters**

- **list_name** (*str*) – List function name
- **view_name** (*str*) – View function name
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **headers** (*dict*) – Additional request headers
- **data** – Request payload
- **params** (*dict*) – Additional request query parameters
- **format** (*str*) – List function output format

For other parameters see `aiocouchdb.v1.designdoc.DesignDocument.view()` method docstring.

**Return type** `HttpResponse`

**name**

Returns a document id specified in class constructor.

**rewrite**(**path*, *auth=None*, *method=None*, *headers=None*, *data=None*, *params=None*)

Requests *rewrite* resource and returns a raw response object.

**Parameters**

- **path** (*str*) – Request path by segments
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **method** (*str*) – HTTP request method
- **headers** (*dict*) – Additional request headers
- **data** – Request payload
- **params** (*dict*) – Additional request query parameters

**Return type** `HttpResponse`

**show**(*show_name*, *docid=None*, ***, *auth=None*, *method=None*, *headers=None*, *data=None*, *params=None*, *format=None*)

Calls a *show function* and returns a raw response object.

**Parameters**

- **show_name** (*str*) – Show function name
- **docid** (*str*) – Document ID
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **method** (*str*) – HTTP request method
- **headers** (*dict*) – Additional request headers
- **data** – Request payload
- **params** (*dict*) – Additional request query parameters
- **format** (*str*) – Show function output format

**Return type** `HttpResponse`

**update**(*update_name*, *docid=None*, ***, *auth=None*, *method=None*, *headers=None*, *data=None*, *params=None*)

Calls a *show function* and returns a raw response object.

**Parameters**

- **update_name** (*str*) – Update function name
- **docid** (*str*) – Document ID
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **method** (*str*) – HTTP request method
- **headers** (*dict*) – Additional request headers
- **data** – Request payload
- **params** (*dict*) – Additional request query parameters

**Return type** `HttpResponse`

**view**(*view_name*, **keys*, *auth=None*, *feed_buffer_size=None*, *att_encoding_info=None*, *attachments=None*, *conflicts=None*, *descending=None*, *endkey=Ellipsis*, *endkey_docid=None*, *group=None*, *group_level=None*, *include_docs=None*, *inclusive_end=None*, *limit=None*, *reduce=None*, *skip=None*, *stale=None*, *startkey=Ellipsis*, *startkey_docid=None*, *update_seq=None*)

Queries a *stored view* by the name with the specified parameters.

**Parameters**

- **view_name** (*str*) – Name of view stored in the related design document
- **keys** (*str*) – List of view index keys to fetch. This method is smart enough to use *GET* or *POST* request depending on amount of `keys`
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **att_encoding_info** (*bool*) – Includes encoding information in an attachment stubs
- **attachments** (*bool*) – Includes attachments content into documents. **Warning**: use with caution!
- **conflicts** (*bool*) – Includes conflicts information into documents
- **descending** (*bool*) – Return rows in descending by key order
- **endkey** – Stop fetching rows when the specified key is reached
- **endkey_docid** (*str*) – Stop fetching rows when the specified document ID is reached
- **group** (*bool*) – Reduces the view result grouping by unique keys
- **group_level** (*int*) – Reduces the view result grouping the keys with defined level
- **include_docs** (*str*) – Include document body for each row
- **inclusive_end** (*bool*) – When `False`, doesn't includes `endkey` in returned rows
- **limit** (*int*) – Limits the number of the returned rows by the specified number
- **reduce** (*bool*) – Defines is the reduce function needs to be applied or not
- **skip** (*int*) – Skips specified number of rows before starting to return the actual result
- **stale** (*str*) – Allow to fetch the rows from a stale view, without triggering index update. Supported values: `ok` and `update_after`

- **startkey** – Return rows starting with the specified key
- **startkey_docid** (*str*) – Return rows starting with the specified document ID
- **update_seq** (*bool*) – Include an `update_seq` value into view results header

**Return type** `aiocouchdb.feeds.ViewFeed`

**view_class**
alias of `View`

## 1.5 Attachment

**class** `aiocouchdb.v1.attachment.`**Attachment**(*url_or_resource*, *, *name=None*)
Implementation of *CouchDB Attachment API*.

**accepts_range**(*rev=None*, *, *auth=None*)
Returns `True` if attachments accepts bytes range requests.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**delete**(*rev*, *, *auth=None*)
Deletes an attachment.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** dict

**exists**(*rev=None*, *, *auth=None*)
Checks if attachment exists. Assumes success on receiving response with *200 OK* status.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**get**(*rev=None*, *, *auth=None*, *range=None*)
Returns an attachment reader object.

**Parameters**

- **rev** (*str*) – Document revision
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **range** (*slice*) – Bytes range. Could be `slice()` or two-element iterable object like `list` etc or just `int()`

**Return type** `AttachmentReader`

**modified**(*digest*, *, *auth=None*)
Checks if attachment was modified by known MD5 digest.

**Parameters**

- **digest** (*bytes*) – Attachment MD5 digest. Optionally, may be passed in base64 encoding form
- **auth** – `aiocouchdb.authn.AuthProvider` instance

**Return type** bool

**update**(*fileobj*, ***, *auth=None*, *content_encoding=None*, *content_type='application/octet-stream'*, *rev=None*)

Attaches a file to document.

**Parameters**

- **fileobj** (*file*) – File object, should be readable
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **content_encoding** (*str*) – Content encoding: `gzip` or `identity`
- **content_type** (*str*) – Attachment `Content-Type` header
- **rev** (*str*) – Document revision

**Return type** dict

**class** `aiocouchdb.v1.attachment.`**AttachmentReader**(*resp*)

Attachment reader implements `io.RawIOBase` interface with the exception that all I/O bound methods are coroutines.

**close**()

Closes attachment reader and underlying connection.

This method has no effect if the attachment is already closed.

**closed**

Return a bool indicating whether object is closed.

**read**(*size=-1*)

Read and return up to n bytes, where *size* is an `int()`.

Returns an empty bytes object on EOF, or None if the object is set not to block and has no data to read.

**readable**()

Return a bool indicating whether object was opened for reading.

**readall**(*size=8192*)

Read until EOF, using multiple `read()` call.

**readline**()

Read and return a line of bytes from the stream.

If limit is specified, at most limit bytes will be read. Limit should be an `int()`.

The line terminator is always `b'\n'` for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

**readlines**(*hint=None*)

Return a list of lines from the stream.

*hint* can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds *hint*.

CHAPTER 2

# Common Base Objects

## 2.1 Client

**class** aiocouchdb.client.**HttpRequest**(*method*, *url*, ***, *params=None*, *headers=None*, *data=None*, *cookies=None*, *files=None*, *auth=None*, *encoding='utf-8'*, *version=HttpVersion(major=1, minor=1)*, *compress=None*, *chunked=None*, *expect100=False*, *loop=None*, *response_class=None*)

aiohttp.client.ClientRequest class with CouchDB specifics.

**DEFAULT_HEADERS = {'ACCEPT': 'application/json', 'CONTENT-TYPE': 'application/json', 'ACCEPT-ENCODING':**

Default HTTP request headers.

**update_body_from_data**(*data*)

Encodes data as JSON if *Content-Type* is *application/json*.

**class** aiocouchdb.client.**HttpResponse**(*method*, *url*, *host=''*, ***, *writer=None*, *continue100=None*)

Deviation from aiohttp.client.ClientResponse class for CouchDB specifics. Prefers FlowControlChunksQueue flow control which fits the best to handle chunked responses.

**flow_control_class**

alias of FlowControlChunksQueue

**maybe_raise_error**()

Raises an HttpErrorException if response status code is greater or equal *400*.

**read**()

Read response payload.

**class** aiocouchdb.client.**HttpStreamResponse**(*method*, *url*, *host=''*, ***, *writer=None*, *continue100=None*)

Like HttpResponse, but uses FlowControlStreamReader to handle nicely large non-chunked data streams.

**class** aiocouchdb.client.**Resource**(*url*, ***, *request_class=None*, *response_class=None*)

HTTP resource representation. Accepts full url as argument.

```
>>> res = Resource('http://localhost:5984')
>>> res
<aiocouchdb.client.Resource(http://localhost:5984) object at ...>
```

Able to construct new Resource instance by assemble base URL and path sections on call:

```
>>> new_res = res('foo', 'bar/baz')
>>> assert new_res is not res
```

```
>>> new_res.url
'http://localhost:5984/foo/bar%2Fbaz'
```

**apply_auth**(*auth_provider*, *url*, *headers*)
Applies authentication routines on further request.

**Parameters**

- **auth_provider** – `aiocouchdb.authn.AuthProvider` instance
- **url** (*str*) – Request URL
- **headers** (*dict*) – Request headers

**copy**(*path=None*, ***options*)
Makes COPY request to the resource. See `Resource.request()` for arguments definition.

**delete**(*path=None*, ***options*)
Makes DELETE request to the resource. See `Resource.request()` for arguments definition.

**get**(*path=None*, ***options*)
Makes GET request to the resource. See `Resource.request()` for arguments definition.

**head**(*path=None*, ***options*)
Makes HEAD request to the resource. See `Resource.request()` for arguments definition.

**options**(*path=None*, ***options*)
Makes OPTIONS request to the resource. See `Resource.request()` for arguments definition.

**post**(*path=None*, ***options*)
Makes POST request to the resource. See `Resource.request()` for arguments definition.

**put**(*path=None*, ***options*)
Makes PUT request to the resource. See `Resource.request()` for arguments definition.

**request**(*method*, *path=None*, *data=None*, *headers=None*, *params=None*, *auth=None*, ***options*)
Makes a HTTP request to the resource.

**Parameters**

- **method** (*str*) – HTTP method
- **path** (*str*) – Resource relative path
- **data** (*bytes*) – POST/PUT request payload data
- **headers** (*dict*) – Custom HTTP request headers
- **params** (*dict*) – Custom HTTP request query parameters
- **auth** – `aiocouchdb.authn.AuthProvider` instance
- **options** – Additional options for `aiohttp.client.request()` function

**Returns** `aiocouchdb.client.HttpResponse` instance

**request_class**
alias of `HttpRequest`

**response_class**
alias of `HttpResponse`

**update_auth**(*auth_provider*, *response*)
Updates authentication provider state from the HTTP response data.

**Parameters**

- **auth_provider** – `aiocouchdb.authn.AuthProvider` instance
- **response** – `aiocouchdb.client.HttpResponse` instance

`aiocouchdb.client.`**`extract_credentials`**(*url*)

Extract authentication (user name and password) credentials from the given URL.

```
>>> extract_credentials('http://localhost:5984/_config/')
('http://localhost:5984/_config/', None)
>>> extract_credentials('http://joe:secret@localhost:5984/_config/')
('http://localhost:5984/_config/', ('joe', 'secret'))
>>> extract_credentials('http://joe%40example.com:secret@localhost:5984/_config/')
('http://localhost:5984/_config/', ('joe@example.com', 'secret'))
```

`aiocouchdb.client.`**`urljoin`**(*base*, **path*)

Assemble a URI based on a base, any number of path segments, and query string parameters.

```
>>> urljoin('http://example.org', '_all_dbs')
'http://example.org/_all_dbs'
```

A trailing slash on the uri base is handled gracefully:

```
>>> urljoin('http://example.org/', '_all_dbs')
'http://example.org/_all_dbs'
```

And multiple positional arguments become path parts:

```
>>> urljoin('http://example.org/', 'foo', 'bar')
'http://example.org/foo/bar'
```

All slashes within a path part are escaped:

```
>>> urljoin('http://example.org/', 'foo/bar')
'http://example.org/foo%2Fbar'
>>> urljoin('http://example.org/', 'foo', '/bar/')
'http://example.org/foo/%2Fbar%2F'

>>> urljoin('http://example.org/', None)
Traceback (most recent call last):
    ...
TypeError: argument 2 to map() must support iteration
```

## 2.2 Authentication Providers

**class** `aiocouchdb.authn.`**`BasicAuthProvider`**(*name=None*, *password=None*)

Provides authentication via BasicAuth method.

**`credentials`**()

Returns authentication credentials.

**Return type** `aiocouchdb.authn.BasicAuthCredentials`

**`reset`**()

Resets provider instance to default state.

**`set_credentials`**(*name*, *password*)

Sets authentication credentials.

**Parameters**

- **name** (*str*) – Username
- **password** (*str*) – User's password

**sign**(*url*, *headers*)
Adds BasicAuth header to `headers`.

**Parameters**

- **url** (*str*) – Request URL
- **headers** (*dict*) – Request headers

**class** aiocouchdb.authn.**BasicAuthCredentials**
BasicAuth credentials

**password**
Alias for field number 1

**username**
Alias for field number 0

**class** aiocouchdb.authn.**CookieAuthProvider**
Provides authentication by cookies.

**reset**()
Resets provider instance to default state.

**sign**(*url*, *headers*)
Adds cookies to provided `headers`. If `headers` already contains any cookies, they would be merged with instance ones.

**Parameters**

- **url** (*str*) – Request URL
- **headers** (*dict*) – Request headers

**update**(*response*)
Updates cookies from the response.

**Parameters** **response** – `aiocouchdb.client.HttpResponse` instance

**class** aiocouchdb.authn.**OAuthProvider**(*, *consumer_key=None*, *consumer_secret=None*, *resource_key=None*, *resource_secret=None*)
Provides authentication via OAuth1. Requires `oauthlib` package.

**credentials**()
Returns OAuth credentials.

**Return type** `aiocouchdb.authn.OAuthCredentials`

**reset**()
Resets provider instance to default state.

**set_credentials**(*, *consumer_key=None*, *consumer_secret=None*, *resource_key=None*, *resource_secret=None*)
Sets OAuth credentials. Currently, all keyword arguments are required for successful auth.

**Parameters**

- **consumer_key** (*str*) – Consumer key (consumer token)
- **consumer_secret** (*str*) – Consumer secret
- **resource_key** (*str*) – Resource key (oauth token)

- **resource_secret** (*str*) – Resource secret (oauth token secret)

**sign**(*url*, *headers*)
Adds OAuth1 signature to `headers`.

**Parameters**

- **url** (*str*) – Request URL
- **headers** (*dict*) – Request headers

**class** aiocouchdb.authn.**OAuthCredentials**
OAuth credentials

**consumer_key**
Alias for field number 0

**consumer_secret**
Alias for field number 1

**resource_key**
Alias for field number 2

**resource_secret**
Alias for field number 3

**class** aiocouchdb.authn.**ProxyAuthProvider**(*username=None*, *roles=None*, *secret=None*, ***, *x_auth_username=None*, *x_auth_roles=None*, *x_auth_token=None*)
Provides CouchDB proxy authentication methods.

**credentials**()
Returns three-element tuple of defined username, roles and secret.

**reset**()
Resets provider instance to default state.

**set_credentials**(*username*, *roles=None*, *secret=None*)
Sets ProxyAuth credentials.

**Parameters**

- **username** (*str*) – CouchDB username
- **roles** (*list*) – List of username roles
- **secret** (*str*) – ProxyAuth secret. Should match the one which defined on target CouchDB server.

**sign**(*url*, *headers*)
Adds ProxyAuth credentials to `headers`.

**Parameters**

- **url** (*str*) – Request URL
- **headers** (*dict*) – Request headers

**x_auth_roles = 'X-AUTH-COUCHDB-ROLES'**
Controls the name of header used to specify list of CouchDB user roles

**x_auth_token = 'X-AUTH-COUCHDB-TOKEN'**
Controls the name of header used to provide authentication token

**x_auth_username = 'X-AUTH-COUCHDB-USERNAME'**
Controls the name of header used to specify CouchDB username

**class** aiocouchdb.authn.**ProxyAuthCredentials**
ProxyAuth credentials

**roles**
Alias for field number 1

**secret**
Alias for field number 2

**username**
Alias for field number 0

## 2.3 Feeds

**class** aiocouchdb.feeds.**Feed**(*resp*, *, *loop=None*, *buffer_size=0*)
Wrapper over HttpResponse content to stream continuous response by emitted chunks.

**buffer_size = 0**
Limits amount of items feed would fetch and keep for further iteration.

**close**(*force=False*)
Closes feed and the related request connection. Closing feed doesnt means that all

**Parameters force** (*bool*) – In case of True, close connection instead of release. See aiohttp.client.ClientResponse.close() for the details

**is_active**()
Checks if the feed is still able to emit any data.

**Return type** bool

**next**()
Emits the next response chunk or None is feed is empty.

**Return type** bytearray

**class** aiocouchdb.feeds.**JsonFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
As Feed, but for chunked JSON response. Assumes that each received chunk is valid JSON object and decodes them before emit.

**next**()
Decodes feed chunk with JSON before emit it.

**Return type** dict

**class** aiocouchdb.feeds.**ViewFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Like JsonFeed, but uses CouchDB view response specifics.

**next**()
Emits view result row.

**Return type** dict

**offset**
Returns view results offset.

**total_rows**
Returns total rows in view.

**update_seq**
Returns update sequence for a view.

**class** aiocouchdb.feeds.**ChangesFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Processes database changes feed.

**last_seq**
Returns last emitted sequence number.

**Return type** int

**next**()
Emits the next event from changes feed.

**Return type** dict

**class** aiocouchdb.feeds.**LongPollChangesFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Processes long polling database changes feed.

**class** aiocouchdb.feeds.**ContinuousChangesFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Processes continuous database changes feed.

**next**()
Emits the next event from changes feed.

**Return type** dict

**class** aiocouchdb.feeds.**EventSourceFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Handles EventSource response following the W3.org spec with single exception: it expects field *data* to contain valid JSON value.

**next**()
Emits decoded EventSource event.

**Return type** dict

**class** aiocouchdb.feeds.**EventSourceChangesFeed**(*resp*, *, *loop=None*, *buffer_size=0*)
Process event source database changes feed. Similar to `EventSourceFeed`, but includes specifics for changes feed and emits events in the same format as others `ChangesFeed` does.

**next**()
Emits the next event from changes feed.

**Return type** dict

## 2.4 Multipart

**class** aiocouchdb.multipart.**MultipartReader**(*headers*, *content*)
Multipart body reader.

**at_eof**()
Returns `True` if the final boundary was reached or `False` otherwise.

**Return type** bool

**fetch_next_part**()
Returns the next body part reader.

**classmethod from_response**(*response*)
Constructs reader instance from HTTP response.

**Parameters** **response** – `HttpResponse` instance

**multipart_reader_cls = None**
Multipart reader class, used to handle multipart/* body parts. None points to type(self)

**next**()
Emits the next multipart body part.

**part_reader_cls**
Body part reader class for non multipart/* content types.

alias of BodyPartReader

**release**()
Reads all the body parts to the void till the final boundary.

**response_wrapper_cls**
Response wrapper, used when multipart readers constructs from response.

alias of MultipartResponseWrapper

**class** aiocouchdb.multipart.**MultipartWriter**(*subtype='mixed'*, *boundary=None*)
Multipart body writer.

**append**(*obj*, *headers=None*)
Adds a new body part to multipart writer.

**append_form**(*obj*, *headers=None*)
Helper to append form urlencoded part.

**append_json**(*obj*, *headers=None*)
Helper to append JSON part.

**part_writer_cls**
Body part reader class for non multipart/* content types.

alias of BodyPartWriter

**serialize**()
Yields multipart byte chunks.

**class** aiocouchdb.multipart.**BodyPartReader**(*boundary*, *headers*, *content*)
Multipart reader for single body part.

**at_eof**()
Returns True if the boundary was reached or False otherwise.

**Return type** bool

**decode**(*data*)
Decodes data according the specified *Content-Encoding* or *Content-Transfer-Encoding* headers value.

Supports gzip, deflate and identity encodings for *Content-Encoding* header.

Supports base64, quoted-printable encodings for *Content-Transfer-Encoding* header.

**Parameters** **data** (*bytearray*) – Data to decode.

**Raises** RuntimeError - if encoding is unknown.

**Return type** bytes

**filename**
Returns filename specified in Content-Disposition header or None if missed or header is malformed.

**form**(*, *encoding=None*)
Lke read(), but assumes that body parts contains form urlencoded data.

**Parameters** **encoding** (*str*) – Custom form encoding. Overrides specified in charset param of *Content-Type* header

**get_charset**(*default=None*)
Returns charset parameter from `Content-Type` header or default.

**json**(*, *encoding=None*)
Lke `read()`, but assumes that body parts contains JSON data.

**Parameters** **encoding** (*str*) – Custom JSON encoding. Overrides specified in charset param of *Content-Type* header

**read**(*, *decode=False*)
Reads body part data.

**Parameters** **decode** (*bool*) – Decodes data following by encoding method from *Content-Encoding* header. If it missed data remains untouched

**Return type** bytearray

**read_chunk**(*size=8192*)
Reads body part content chunk of the specified size. The body part must has *Content-Length* header with proper value.

**Parameters** **size** (*int*) – chunk size

**Return type** bytearray

**readline**()
Reads body part by line by line.

**Return type** bytearray

**release**()
Lke `read()`, but reads all the data to the void.

**Return type** None

**text**(*, *encoding=None*)
Lke `read()`, but assumes that body part contains text data.

**Parameters** **encoding** (*str*) – Custom text encoding. Overrides specified in charset param of *Content-Type* header

**Return type** str

**class** aiocouchdb.multipart.**BodyPartWriter**(*obj*, *headers=None*, *, *chunk_size=8192*)
Multipart writer for single body part.

**filename**
Returns filename specified in Content-Disposition header or `None` if missed.

**serialize**()
Yields byte chunks for body part.

**set_content_disposition**(*disptype*, ***params*)
Sets `Content-Disposition` header.

**Parameters**

- **disptype** (*str*) – Disposition type: inline, attachment, form-data. Should be valid extension token (see RFC 2183)
- **params** (*dict*) – Disposition params

## 2.5 Views

**class** aiocouchdb.views.**View**(*resource*)
Views requesting helper.

**request**(*, *auth=None*, *feed_buffer_size=None*, *data=None*, *params=None*)
Requests a view associated with the owned resource.

**Parameters**

- **auth** – aiocouchdb.authn.AuthProvider instance
- **feed_buffer_size** (*int*) – Internal buffer size for fetched feed items
- **data** (*dict*) – View request payload
- **params** (*dict*) – View request query parameters

**Return type** aiocouchdb.feeds.ViewFeed

## 2.6 Errors

### 2.6.1 Exception hierarchy

```
BaseException
+-- Exception
    +-- aiohttp.errors.HttpProcessingError
        +-- aiocouchdb.errors.HttpErrorException
            +-- aiocouchdb.errors.BadRequest
            +-- aiocouchdb.errors.Unauthorized
            +-- aiocouchdb.errors.Forbidden
            +-- aiocouchdb.errors.ResourceNotFound
            +-- aiocouchdb.errors.MethodNotAllowed
            +-- aiocouchdb.errors.ResourceConflict
            +-- aiocouchdb.errors.PreconditionFailed
            +-- aiocouchdb.errors.RequestedRangeNotSatisfiable
            +-- aiocouchdb.errors.ServerError
```

**exception** aiocouchdb.errors.**HttpErrorException**(*error*, *reason*, *headers=None*)
Extension of aiohttp.errors.HttpErrorException for CouchDB related errors.

**exception** aiocouchdb.errors.**BadRequest**(*error*, *reason*, *headers=None*)
The request could not be understood by the server due to malformed syntax.

**exception** aiocouchdb.errors.**Unauthorized**(*error*, *reason*, *headers=None*)
The request requires user authentication.

**exception** aiocouchdb.errors.**Forbidden**(*error*, *reason*, *headers=None*)
The server understood the request, but is refusing to fulfill it.

**exception** aiocouchdb.errors.**ResourceNotFound**(*error*, *reason*, *headers=None*)
The server has not found anything matching the Request-URI.

**exception** aiocouchdb.errors.**MethodNotAllowed**(*error*, *reason*, *headers=None*)
The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.

**exception** aiocouchdb.errors.**ResourceConflict**(*error*, *reason*, *headers=None*)
The request could not be completed due to a conflict with the current state of the resource.

**exception** aiocouchdb.errors.**PreconditionFailed**(*error*, *reason*, *headers=None*)
The precondition given in one or more of the Request-Header fields evaluated to false when it was tested on the server.

**exception** aiocouchdb.errors.**RequestedRangeNotSatisfiable**(*error*, *reason*, *headers=None*)
The client has asked for a portion of the file, but the server cannot supply that portion.

**exception** aiocouchdb.errors.**ServerError**(*error*, *reason*, *headers=None*)
The server encountered an unexpected condition which prevented it from fulfilling the request.

aiocouchdb.errors.**maybe_raise_error**(*resp*)
Raises aiohttp.errors.HttpErrorException exception in case of >=400 response status code.

## 2.7 Headers

aiocouchdb.hdrs.**ACCEPT_RANGES** = **'ACCEPT-RANGES'**
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.5.html

aiocouchdb.hdrs.**CONTENT_DISPOSITION** = **'CONTENT-DISPOSITION'**
http://tools.ietf.org/html/rfc2183

aiocouchdb.hdrs.**X_AUTH_COUCHDB_ROLES** = **'X-AUTH-COUCHDB-ROLES'**
Defines CouchDB Proxy Auth list of roles separated by a comma

aiocouchdb.hdrs.**X_AUTH_COUCHDB_TOKEN** = **'X-AUTH-COUCHDB-TOKEN'**
Defines CouchDB Proxy Auth token

aiocouchdb.hdrs.**X_AUTH_COUCHDB_USERNAME** = **'X-AUTH-COUCHDB-USERNAME'**
Defines CouchDB Proxy Auth username

CHAPTER 3

# Getting started

**Contents**



If you'd some background experience with couchdb-python client, you'll find *aiocouchdb* API a bit familiar. That project is my lovely too, but suddenly it's completely synchronous.

At first, you need to create instance of `Server` object which interacts with CouchDB Server API:

```
>>> import aiocouchdb
>>> server = aiocouchdb.Server()
>>> server
<aiocouchdb.v1.server.Server(http://localhost:5984) object at 0x7f8199a80350>
```

As like as couchdb-python Server instance has `resource` attribute that acts very familiar:

```
>>> server.resource
<aiocouchdb.client.Resource(http://localhost:5984) object at 0x7f9cba5e2490>
>>> server.resource('db', 'doc1')
<aiocouchdb.client.Resource(http://localhost:5984/db/doc1) object at 0x7f9cb9fc2e10>
```

With the only exception that it's a coroutine:

---

**Note:** Python doesn't supports `yield from` in shell, so examples below are a bit out of a real, but be sure - that's how they works in real.

---

```
>>> resp = yield from server.resource.get()
<ClientResponse(http://localhost:5984) [200 OK]>
<CIMultiDictProxy {'SERVER': 'CouchDB/1.6.1 (Erlang OTP/17)', 'DATE': 'Sun, 08 Mar 2015 15:13:12 GMT'
>>> yield from resp.json()
{'couchdb': 'Welcome!',
 'vendor': {'version': '1.6.1', 'name': 'The Apache Software Foundation'},
 'uuid': '0510c29b75ae33fd3975eb505db2dd12',
 'version': '1.6.1'}
```

The `Resource` object provides a tiny wrapper over `aiohttp.client.request()` function so you can use it in case of raw API access.

But, libraries are made to hide all the implementation details and make work with API nice and easy one and *aiocouchdb* isn't an exception. The example above is actually what the `Server.info()` method does:

```
>>> yield from server.info()
{'couchdb': 'Welcome!',
 'vendor': {'version': '1.6.1', 'name': 'The Apache Software Foundation'},
 'uuid': '0510c29b75ae33fd3975eb505db2dd12',
 'version': '1.6.1'}
```

Most of `Server` and not only methods are named similar to the real HTTP API endpoints:

```
>>> yield from server.all_dbs()
['_replicator', '_users', 'db']
>>> yield from server.active_tasks()
[{'database': 'db',
  'pid': '<0.10209.20>',
  'changes_done': 0,
  'progress': 0,
  'started_on': 1425805499,
  'total_changes': 1430818,
  'type': 'database_compaction',
  'updated_on': 1425805499}]
```

With a few exceptions like `Server.session` or `Server.config` which has complex use-case behind and are operated by other objects.

Speaking about `aiocouchdb.v1.server.Server.session`, *aiocouchdb* supports *multiuser workflow* where you pass session object as an argument on resource request.

```
>>> admin = yield from server.session.open('admin', 's3cr1t')
>>> user = yield from server.session.open('user', 'pass')
```

Here we just opened two session for different users. Their usage is pretty trivial - just pass them as `auth` keyword parameter to every API function call:

```
>>> yield from server.active_tasks(auth=admin)
[{'database': 'db',
  'pid': '<0.10209.20>',
  'changes_done': 50413,
  'progress': 3,
  'started_on': 1425805499,
  'total_changes': 1430818,
  'type': 'database_compaction',
  'updated_on': 1425806018}]
>>> yield from server.active_tasks(auth=user)
Traceback:
...
Unauthorized: [forbidden] You are not a server admin.
```

Another important moment that *aiocouchdb* raises exception on HTTP errors. By using `Resource` object you'll receive raw response and may build custom logic on processing such errors: to raise an exception or to not.

With using `Server.session.open()` you implicitly creates `CookieAuthProvider` which hold received from CouchDB cookie with authentication token. *aiocouchdb* also provides the way to authorize via *Basic Auth*, *OAuth* (oauthlib required) and others. Their usage is also pretty trivial:

```
>>> admin = aiocouchdb.BasicAuthProvider('admin', 's3cr1t')
>>> yield from server.active_tasks(auth=admin)
[{'database': 'db',
  'pid': '<0.10209.20>',
  'changes_done': 50413,
  'progress': 3,
  'started_on': 1425805499,
  'total_changes': 1430818,
  'type': 'database_compaction',
  'updated_on': 1425806018}]
```

CHAPTER 4

# Working with databases

To create a database object which will interact with CouchDB Database API you have three ways to go:

1. Using direct object instance creation:

```
>>> aiocouchdb.Database('http://localhost:5984/db')
<aiocouchdb.v1.database.Database(http://localhost:5984/db) object at 0x7ffd44d58f90>
```

2. Using `__getitem__` protocol similar to couchdb-python:

```
>>> server['db']
<aiocouchdb.v1.database.Database(http://localhost:5984/db) object at 0x7ffd44cf30d0>
```

3. Using `Server.db()` method:

```
>>> yield from server.db('db')
<aiocouchdb.v1.database.Database(http://localhost:5984/db) object at 0x7ffd44cf3390>
```

What's their difference? First method is useful when you don't have access to a `Server` instance, but knows database URL. Second one returns instantly a `Database` instance for the name you specified.

But the third one is smarter: it verifies that database by name you'd specified is accessible for you and if it's not - raises an exception:

```
>>> yield from server.db('_users')
Traceback:
...
Unauthorized: [forbidden] You are not a server admin.
>>> yield from server.db('_foo')
Traceback:
...
BadRequest: [illegal_database_name] Name: '_foo'. Only lowercase characters (a-z), digits (0-9), and
```

This costs you an additional HTTP request, but gives the insurance that the following methods calls will not fail by unrelated reasons.

This method doesn't raises an exception if database doesn't exists to allow you create it:

```
>>> db = yield from server.db('newdb')
>>> yield from db.exists()
False
>>> yield from db.create()
True
>>> yield from db.exists()
True
```

## 4.1 Iterating over documents

In couchdb-python you might done it with in the following way:

```
>>> for docid in db:
...     do_something(db[docid])
```

Or:

```
>>> for row in db.view('_all_docs'):
...     do_something(db[row['id']])
```

*aiocouchdb* does that quite differently:

```
>>> res = yield from db.all_docs()
>>> while True:
...     rec = yield from res.next()
...     if rec is None:
...         break
...     do_something(rec['id'])
```

What's going on here?

1. You requesting */db/_all_docs* endpoint explicitly and may pass all his query parameters as you need;
2. On `Database.all_docs()` call returns not a list of view results, but a special instance of `ViewFeed` object which fetches results one by one in background into internal buffer without loading whole result into memory in single shot. You can control this buffer size with *feed_buffer_size* keyword argument;
3. When all the records are processed it emits None which signs on empty feed and the loop breaking out;

*aiocouchdb* tries never load large streams, but process them in iterative way. This may looks ugly for small data sets, but when you deal with the large ones it'll save you a lot of resources.

The same loop pattern in used to process `Database.changes()` as well.

CHAPTER 5

# Working with documents

To work with a document you need get `Document` instance first - `Database` doesn't knows anything about CouchDB Document API. The way to do this is the same as for database:

```
>>> aiocouchdb.Document('http://localhost:5984/db/doc1')
<aiocouchdb.v1.document.Document(http://localhost:5984/db/doc1) object at 0x7ff3ef7af070>
>>> server['db']['doc1']
<aiocouchdb.v1.document.Document(http://localhost:5984/db/doc1) object at 0x7fda92ff4850>
>>> doc = yield from db.doc('doc1')
<aiocouchdb.v1.document.Document(http://localhost:5984/db/doc1) object at 0x7fda981380d0>
```

Their difference is the same as for `Database` mentioned above.

```
>>> yield from doc.exists()
False
>>> meta = yield from doc.update({'hello': 'CouchDB'})
>>> meta
{'ok': True, 'rev': '1-7c6fb984afda7e07d030cce000dc5965', 'id': 'doc1'}
>>> yield from doc.exists()
True
>>> meta = yield from doc.update({'hello': 'CouchDB'}, rev=meta['rev'])
>>> meta
{'ok': True, 'rev': '2-c5298951d02b03f3d6273ad5854ea729', 'id': 'doc1'}
>>> yield from doc.get()
{'hello': 'CouchDB',
 '_id': 'doc1',
 '_rev': '2-c5298951d02b03f3d6273ad5854ea729'}
>>> yield from doc.delete('2-c5298951d02b03f3d6273ad5854ea729')
{'ok': True, 'rev': '3-cfa05c76fb4a0557605d6a8b1a765055', 'id': 'doc1'}
>>> yield from doc.exists()
False
```

Pretty simple, right?

CHAPTER 6

# What's next?

There are a lot of things are left untold. Checkout *CouchDB 1x* API for more. Happy hacking!

CHAPTER 7

# Changes

## 7.1 0.8.0 (2015-03-20)

- Source tree was refactored in the way to support multiple major CouchDB versions as like as the other friendly forks
- Database create and delete methods now return exact the same response as CouchDB sends back
- Each module now contains __all__ list to normalize their exports
- API classes and Resource now has nicer __repr__ output
- Better error messages format
- Fix function_clause error on attempt to update a document with attachments by using multipart request
- Document.update doesn't makes document's dict invalid for further requests after multipart one
- Fixed accidental payload sent with HEAD/GET/DELETE requests which caused connection close from CouchDB side
- Added integration with Travis CI
- Code cleaned by following pylint and flake8 notices
- Added short tutorial for documentation
- Minor fixes and Makefile improvements

## 7.2 0.7.0 (2015-02-18)

- Greatly improved multipart module, added multipart writer
- Document.update now supports multipart requests to upload multiple attachments in single request
- Added Proxy Authentication provider
- Minimal requirements for aiohttp raised up to 0.14.0 version

## 7.3 0.6.0 (2014-11-12)

- Adopt test suite to run against real CouchDB instance
- Database, documents and attachments now provides access to their name/id

- Remove redundant longnamed constructors
- Construct Database/Document/Attachment instances through __getitem__ protocol
- Add Document.rev method to get current document's revision
- Add helpers to work with authentication database (_users)
- Add optional limitation of feeds buffer
- All remove(...) methods are renamed to delete(...) ones
- Add support for config option existence check
- Correctly set members for database security
- Fix requests with Accept-Ranges header against attachments
- Fix views requests when startkey/endkey should be null
- Allow to pass custom query parameters and request headers onto changes feed request
- Handle correctly HTTP 416 error response
- Minor code fixes and cleanup

## 7.4 0.5.0 (2014-09-26)

- Last checkpoint release. It's in beta now!
- Implements CouchDB Design Documents HTTP API
- Views refactoring and implementation consolidation

## 7.5 0.4.0 (2014-09-17)

- Another checkpoint release
- Implements CouchDB Attachment HTTP API
- Minimal requirements for aiohttp raised up to 0.9.1 version
- Minor fixes for Document API

## 7.6 0.3.0 (2014-08-18)

- Third checkpoint release
- Implements CouchDB Document HTTP API
- Support document's multipart API (but not doc update due to COUCHDB-2295)
- Minimal requirements for aiohttp raised up to 0.9.0 version
- Better documentation

## 7.7 0.2.0 (2014-07-08)

- Second checkpoint release
- Implements CouchDB Database HTTP API
- Bulk docs accepts generator as an argument and streams request doc by doc
- Views are processed as stream
- Unified output for various changes feed types
- Basic Auth accepts non-ASCII credentials
- Minimal requirements for aiohttp raised up to 0.8.4 version

## 7.8 0.1.0 (2014-07-01)

- Initial checkpoint release
- Implements CouchDB Server HTTP API
- BasicAuth, Cookie, OAuth authentication providers
- Multi-session workflow

CHAPTER 8

# License

## a

## A



## B



## C



## D

## E



## F



## G



## H



## I



## J



## L



## M

## N



## O



## P



## R

## S



## T



## U



## V



## X