

# Package ‘R4CouchDB’

March 1, 2017

**Type** Package

**Title** A R Convenience Layer for CouchDB 2.0

**Version** 0.7.5

**Date** 2017-02-25

**Author** Thomas Bock

**URL** <https://github.com/wactbprot/R4CouchDB>

**Maintainer** Thomas Bock <thsteinbock@web.de>

**Description** Provides a collection of functions for basic database and document management operations such as add, get, list access or delete. Every cdbFunction() gets and returns a list() containing the connection setup. Such a list can be generated by cdbIni().

**License** MIT + file LICENSE

**LazyLoad** yes

**Depends** R (>= 2.7.0), bitops, RCurl (>= 1.95), RJSONIO (>= 1.3)

**Suggests** roxygen2 (>= 4.0), testthat (>= 0.8)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-03-01 19:05:42

## R topics documented:

cdbAddAttachment . . . . .	2
cdbAddDoc . . . . .	3
cdbAddDocS . . . . .	4
cdbDeleteDoc . . . . .	5
cdbGetConfig . . . . .	6
cdbGetDoc . . . . .	6
cdbGetList . . . . .	7
cdbGetShow . . . . .	8
cdbGetUuid . . . . .	9
cdbGetUuidS . . . . .	10

cdbGetView . . . . .	11
cdbIni . . . . .	11
cdbListDB . . . . .	13
cdbMakeDB . . . . .	14
cdbRemoveDB . . . . .	15
cdbUpdateDoc . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

cdbAddAttachment	<i>Add attachments</i>
------------------	------------------------

---

### Description

This function adds attachments to a database document that already exists.

### Usage

```
cdbAddAttachment(cdb)
```

### Arguments

cdb	The list cdb has to contain cdb\$fileName, cdb\$serverName, cdb\$DBName and a cdb\$id.
-----	--

### Details

The function uses the Rcurl- function guessMIMEtype() to do exactly this: guessing the mime type of cdb\$fileName.

If the switch cdb\$attachmentsWithPath is set to TRUE the attachment is saved with the path. This behavior is default since version 0.2.5 of R4CouchDB

### Value

cdb	The result is stored in cdb\$res
-----	----------------------------------

### Author(s)

wactbprot

### Examples

```
## Not run:
ccc      <- cdbIni(DBName="r4couch_db")
ccc$dataList <- list(normalDistRand = rnorm(20))
ccc      <- cdbAddDoc(ccc)
# make a 3d plot (stolen from ?persp)
x        <- seq(-10, 10, length= 30)
y        <- x
```

```

f          <- function(x,y) {r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z          <- outer(x, y, f)

z[is.na(z)] <- 1
op         <- par(bg = "black")
ccc$fileName <- "3dplot.pdf"

pdf(ccc$fileName)
persp(x, y, z,
      theta = 30,
      phi = 30,
      expand = 0.5,
      col = "lightblue")
dev.off()
# add the plot as attachment to the database
# it workes over ccc$fileName
ccc        <- cdbAddAttachment(ccc)

## End(Not run)

```

---

 cdbAddDoc

*Generates a new document*


---

## Description

This function adds a new document to an already existing database

## Usage

```
cdbAddDoc(cdb)
```

## Arguments

cdb                    The list cdb only has to contain a cdb\$dataList which is not an empty list().

## Details

This function is called addDoc (which means add a new document). Therefore the cdb\$id is requested using cdbGetUuid() for every document to add if no cdb\$id is provided. If a cdb\$id is provided the function fails when a document with the given id already exists. In this case one should use cdbUpdateDoc(). Since version v0.6 the function writes the \_rev and \_id key to the top level of cdb\$dataList.

## Value

cdb                    The couchdb response is stored in cdb\$res

## Author(s)

wactbprot

**See Also**

`cdbGetDoc()`

**Examples**

```
## Not run:
ccc          <- cdbIni()
# I assume a database at localhost:5984 already exists
ccc$DBName   <- "r4couchdb_db"
ccc$dataList <- list(normalDistRand = rnorm(20))
ccc          <- cdbAddDoc(ccc)

## End(Not run)
```

---

`cdbAddDocS`

*This function adds multiple database documents with one request*

---

**Description**

This is done via the `_bulk_docs` API provided by an already existing database.

**Usage**

```
cdbAddDocS(cdb)
```

**Arguments**

`cdb`                    `cdb$dataList` has to be a list of lists, `cdb$DBName`, `cdb$serverName` is needed.

**Details**

The `_bulk_docs` endpoint requires that `cdb$dataList` resolves to an json array. This is reached with e.g. `cdb$dataList <- list(list(...),list(...),...)`. Furthermore, `_bulk_docs` requires the documents to be wrapped in a key named `docs:[...]`; this is done by `cdbAddDocS()` if `cdb$dataList` is a list of lists. The user dont need to care.

At the moment the resulting `_rev` and `_id` will be not written back to the `cdb$dataList`. This means that a second call of `cdbAddDocS()` generates new Documents.

**Value**

`cdb`                    The couchdb response is stored in `cdb$res`

**Author(s)**

parisni, wactbprot

**See Also**

`cdbAddDoc()`

**Examples**

```
## Not run:
ccc          <- cdbIni()
# I assume a database at localhost:5984 already exists
ccc$DBName   <- "r4couchdb_db"
docs <- list()
for(i in 1:10){
  docs[[i]] <- list(normalDistRand = rnorm(20))
}
# docs is noe a list of 10 lists
ccc$dataList <- docs
# generating 10 database documents
cccAddDocS(ccc)$res

## End(Not run)
```

---

`cdbDeleteDoc`

*Deletes a document from a database*

---

**Description**

With a given `cdb$id` this function sends a http "DELETE" request to the url `.../cdb$id?rev=cdb$rev`.

**Usage**

`cdbDeleteDoc(cdb)`

**Arguments**

`cdb`                    Beside `cdb$serverName`, `cdb$port` and `cdb$DBName` the `cdb$id` must be given.  
R errors are reported in `cdb$errors`

**Value**

`cdb`                    The result of the delete request is stored in `cdb$res`(whatever this means).

**Author(s)**

wactbprot

**See Also**

`cdbAddDoc()`

cdbGetConfig

*Request couchdb config*

---

**Description**

Function provides access to the `_config` api end point.

**Usage**

```
cdbGetConfig(cdb)
```

**Arguments**

`cdb` Only the connection settings `cdb$port` and `cdb$serverName` is needed.

**Value**

`cdb` The result of the request is stored in `cdb$res` after converting the answer into a list using `fromJSON()`.

**Author(s)**

wactbprot

**See Also**

`cdbMakeDB`

**Examples**

```
## Not run:  
cdbGetConfig(cdbIni())$res  
  
## End(Not run)
```

---

cdbGetDoc*Get a doc from CouchDB*

---

**Description**

With a given `cdb$id` this function requests the document.

**Usage**

```
cdbGetDoc(cdb)
```

**Arguments**

cdb                    Beside cdb\$serverName, cdb\$port and cdb\$DBName the cdb\$id must be given.  
R errors are reported  
in cdb\$errors

**Value**

cdb                    The result of the request is stored in cdb\$res after converting the answer into a  
list using fromJSON(). If a list entry needed in cdb is not provided cdb\$error  
gives some information.

**Author(s)**

wactbprot

**See Also**

cdbAddDoc()

**Examples**

```
## Not run:
ccc                    <- cdbIni()
ccc$newDBName         <- "r4couchdb_db"
ccc$dataList         <- list(normalDistRand = rnorm(20))
ccc                    <- cdbAddDoc(ccc)
cdbGetDoc(ccc)$res

## End(Not run)
```

---

cdbGetList

*Receive list results from CouchDB*

---

**Description**

The function provides accesses to CouchDB lists.

**Usage**

```
cdbGetList(cdb)
```

**Arguments**

cdb                    Beside the connection details (cdb\$port, cdb\$DBName ...) the cdb\$design cdb\$view  
and cdb\$list is needed.

**Details**

Query params e.g. "reduce=false" or "group\_level=1" can be provided in cdb\$queryParam. By now multiple params must be given in one string e.g. "a=b&c=d&e=f".

**Value**

cdb                    The result of the request is stored in cdb\$res after converting the json answer into a list using cdb\$fromJSON(). If a needed cdb (design, list, view, ...) entry was not provided cdb\$error says something about the R side.

**Author(s)**

wactbprot

---

cdbGetShow

*Receive show results from CouchDB*

---

**Description**

The function provides accesses to CouchDB shows.

**Usage**

```
cdbGetShow(cdb)
```

**Arguments**

cdb                    Beside the connection details (cdb\$port, cdb\$DBName ...) the cdb\$design and cdb\$show is needed.

**Details**

Query params e.g. "format=json" can be provided in cdb\$queryParam. Multiple params must be given in one string e.g. "a=b&c=d&e=f".

**Value**

cdb                    The result of the request is stored in cdb\$res after converting the json answer into a list using cdb\$fromJSON(). If a needed cdb entry was not provided cdb\$error provides information.

**Author(s)**

wactbprot



---

cdbGetUuid	<i>Function for request one id</i>
------------	------------------------------------

---

**Description**

Function returns a 128bit uuid requested from CouchDB

**Usage**

```
cdbGetUuid(cdb)
```

**Arguments**

cdb	Only the connection settings cdb\$port and cdb\$serverName is needed.
-----	---

**Details**

Simple CouchDB API end point to `http://serverName:port/_uuids`.

**Value**

cdb	The result of the request is stored in cdb\$id after converting the answer into a list using <code>fromJSON()</code> .
-----	--

**Author(s)**

wactbprot

**See Also**

cdbMakeDB

**Examples**

```
## Not run:  
cdbGetUuid(cdbIni())$res  
  
## End(Not run)
```

cdbGetUuidS

*Function for request some ids*

---

**Description**

Function returns a 128bit uuid requested from CouchDB

**Usage**

```
cdbGetUuidS(cdb)
```

**Arguments**

cdb	Only the connection settings cdb\$port, cdb\$serverName and cdb\$count is needed.
-----	---

**Details**

CouchDB API provides the url `http://serverName:port/_uuids` for those clients who aren't able to create those ids. The number N of ids received from a CouchDB can be set by `cdb$count <- N` since version 0.6. The function writes to `cdb$res` (in opposite to `cdbGetUuid()` which writes to `cdb$id`)

**Value**

cdb	The result of the request is stored in <code>cdb\$res</code> after converting the answer into a list using <code>fromJSON()</code> .
-----	--

**Author(s)**

wactbprot

**See Also**

`cdbMakeDB`

**Examples**

```
## Not run:
ccc      <- cdbIni()
ccc$count <- 100
cdbGetUuidS(ccc)$res

## End(Not run)
```

---

`cdbGetView`*Receive view results from CouchDB*

---

**Description**

The function provides accesses to CouchDB views.

**Usage**

```
cdbGetView(cdb)
```

**Arguments**

<code>cdb</code>	Beside the connection details ( <code>cdb\$port</code> , <code>cdb\$DName</code> ...) the <code>cdb\$design</code> and <code>cdb\$view</code> is needed.
------------------	--

**Details**

Query params e.g. "reduce=false" or "group\_level=1" can be provided in `cdb$queryParam`

**Value**

<code>cdb</code>	The result of the request is stored in <code>cdb\$res</code> after converting the json answer into a list using <code>fromJSON()</code> . If a needed <code>cdb</code> list entry was not provided <code>cdb\$error</code> says something about the R side
------------------	--

**Note**

For the moment only one `cdb$queryParam` is possible. In the future maybe Duncans RJavaScript package can be used to generate views without leaving R.

**Author(s)**

wactbprot

---

`cdbIni`*Ini function*

---

**Description**

Function returns a list with some default settings and often used functions such as `cdb$baseUrl`.

**Usage**

```

cdbIni(serverName="localhost",
port="5984",
prot = "http",
DBName="",
uname = "",
pwd = "",
newDBName = "",
removeDBName = "",
id = "",
fileName = "",
design = "",
view = "",
list = "",
show = "",
queryParam = "",
encSub = "?",
count = 10,
dataList = list(),
attachmentsWithPath=TRUE,
digits = 7)

```

**Arguments**

serverName	server name
port	port
prot	name of the protocol default is http
DBName	name of database
uname	name of the user
pwd	password
newDBName	name of the database for cdbMakeDB()
removeDBName	name of the database to remove with cdbRemoveDB()
id	the document id to get, put, post or delete
fileName	for use in cdbAddAttachment
design	the name of the design used when asking a view or list
view	the name of a view to query
list	the name of a list to query
show	the name of a show to query
queryParam	additional query params
encSub	a character which is used as a replacement for chars who can not be converted by iconv
count	how many uuids should be returned by cdbGetUuidS()
dataList	a list containing data to post or update

`attachmentsWithPath` effects the result of the function `cdbAddAttachment` in the way the variable is named  
`digits` digits kept at toJSON conversion

### Details

The list: `cdb <- list(serverName = "localhost", ... )` is returned if the packages `library(RCurl)` and `library(RJSONIO)` are successfully loaded.

### Value

`cdb` The R4CouchDB (method) chain(ing) list

### Author(s)

wactbprot, parisni

### Examples

```
## Not run:
ccc <- cdbIni(digits=13,
              DBName="r4couch_db",
              attachmentsWithPath=FALSE,
              dataList=list(normalDistRand = rnorm(20)))

## End(Not run)
```

---

`cdbListDB`

*Returns all databases on the server*

---

### Description

Gives a list of all databases available at `cdb$serverName`.

### Usage

```
cdbListDB(cdb)
```

### Arguments

`cdb` Only the connection settings `cdb$port` and `cdb$serverName` is needed.

### Details

The function uses the `_all_dbs` API end point .

**Value**

cdb                    The result of the request is stored in cdb\$res after converting the json answer into a list using cdb\$fromJSON().

**Author(s)**

wactbprot

**See Also**

cdbMakeDB

**Examples**

```
## Not run:
cdbListDB(cdbIni())$res

## End(Not run)
```

---

cdbMakeDB	<i>Creates a new database</i>
-----------	-------------------------------

---

**Description**

The name of the new database is taken from cdb\$newDBName.

**Usage**

```
cdbMakeDB(cdb)
```

**Arguments**

cdb                    The cdb have to provide cdb\$serverName, cdb\$port and cdb\$newDBName

**Details**

The work is done by getURL() from Duncans RCurl package.

After creating the new database the function makes the shortcut cdb\$DBName <- cdb\$newDBName so that further operations happen on the new created database. Finally cdb\$newDBName <- "".

**Value**

cdb                    The CouchDB answer is stored in cdb\$res. Any problems on the R side are reported in cdb\$error

**Note**

The convention for database naming should be implemented.

**Author(s)**

wactbprot

**See Also**

cdbUpdateDoc

**Examples**

```
## Not run:
ccc          <- cdbIni()
ccc$newDBName <- "r4couchdb_db"
ccc          <- cdbMakeDB(ccc)
ccc$res
ccc$removeDBName <- ccc$DBName
cdbRemoveDB(ccc)$res

## End(Not run)
```

---

 cdbRemoveDB

*Function to remove a database*


---

**Description**

Removing a database means sending a http- "DELETE"- request to `http://cdb$serverName:cdb$port/ ...`

**Usage**

```
cdbRemoveDB(cdb)
```

**Arguments**

cdb                    The cdb has to provide cdb\$serverName, cdb\$port and cdb\$DBName

**Details**

In cdb a entry cdb\$delDBName should be provided for more explicit deleting respectively more secure removing.

**Value**

cdb                    The CouchDB answer is stored in cdb\$res. Any problems on the R side are reportet in cdb\$error

**Author(s)**

wactbprot

**See Also**

`cdbMakeDB`

**Examples**

```
## Not run:
ccc          <- cdbIni()
ccc$newDBName <- "r4couchdb_db"
ccc         <- cdbMakeDB(ccc)
ccc$res
ccc$removeDBName <- ccc$DBName
cdbRemoveDB(ccc)$res

## End(Not run)
```

---

`cdbUpdateDoc`

*This function updates an existing doc*

---

**Description**

This essentially means that a revision, corresponding to the '\_id' has to be provided. If no '\_rev' is given in the cdb list the function gets the doc from the db and takes the rev number for the update

**Usage**

```
cdbUpdateDoc(cdb)
```

**Arguments**

`cdb` the cdb connection configuration list must contain the `cdb$serverName`, `cdb$port`, `cdb$DBName` and `cdb$id`. The data which updates the data stored in the doc is provided in `cdb$dataList`

**Details**

Updating a doc at couchdb means executing a http "PUT" request. The cdb list must contain the `cdb$serverName`, `cdb$port`, `cdb$DBName`, `cdb$id`. Since v0.6 the revision of the document should exist at the intended place: `cdb$dataList$'_rev'`.

`getURL()` with `customrequest = "PUT"` does the work. If a needed cdb\$ list entry is not provided `cdb$error` maybe says something about the R side.

**Value**

`cdb` The response of the request is stored in `cdb$res` after converting the answer by means of `fromJSON()`. The revision provided by the respons is used for updating the `cdb$dataList$'_rev'`.



**Author(s)**

wactbprot

**See Also**

`cdbInit()`

**Examples**

```
## Not run:
ccc          <- cdbIni()
# I assume a database at localhost:5984 already exists
ccc$DBName   <- "r4couchdb_db"
ccc$dataList <- list(normalDistRand = rnorm(20))
ccc          <- cdbAddDoc(ccc)

ccc$dataList$Date <- date()
ccc               <- cdbUpdateDoc(ccc)

## End(Not run)
```

# Index

## \*Topic **misc**

- cdbAddAttachment, 2
- cdbAddDoc, 3
- cdbAddDocS, 4
- cdbDeleteDoc, 5
- cdbGetConfig, 6
- cdbGetDoc, 6
- cdbGetList, 7
- cdbGetShow, 8
- cdbGetUuid, 9
- cdbGetUuidS, 10
- cdbGetView, 11
- cdbIni, 11
- cdbListDB, 13
- cdbMakeDB, 14
- cdbRemoveDB, 15
- cdbUpdateDoc, 16

- cdbAddAttachment, 2
- cdbAddDoc, 3
- cdbAddDocS, 4
- cdbDeleteDoc, 5
- cdbGetConfig, 6
- cdbGetDoc, 6
- cdbGetList, 7
- cdbGetShow, 8
- cdbGetUuid, 9
- cdbGetUuidS, 10
- cdbGetView, 11
- cdbIni, 11
- cdbListDB, 13
- cdbMakeDB, 14
- cdbRemoveDB, 15
- cdbUpdateDoc, 16