

---

# **clashcallerbot-reddit Documentation**

***Release 2.1.4***

**Jose A. Lerma III**

**Nov 04, 2018**



**CONTENTS:**

<b>1</b>	<b>clashcallerbot-reddit</b>	<b>1</b>
1.1	clashcallerbot_database module . . . . .	1
1.2	clashcallerbot_reply module . . . . .	4
1.3	clashcallerbot_search module . . . . .	5
1.4	logging_conf module . . . . .	6
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## CLASHCALLERBOT-REDDIT

## 1.1 clashcallerbot\_database module

Setup the MySQL-compatible database.

If run directly, this module will setup the clashcallerbot database with tables and display their format and contents. Additionally, this module provides a class with various methods for managing the MySQL-compatible database:

- Create database and tables.
- View table data and properties.
- Grant user permissions (if logged into database as root).
- Add rows to tables.
- Delete tables and rows.
- Convert python datetime to MySQL datetime.

```
class clashcallerbot_database.ClashCallerDatabase (config_file=None,
                                                    root_user=None)
```

Bases: `object`

Implements a class for a ClashCaller Database.

**config\_file**

A configparser object with database.ini file pre-read.

**Type** `configparser.ConfigParser()`

**root\_user**

Specifies whether the database will be setup as root user.

**Type** `bool`

**mysql\_connection**

A `mysql.connector.connect()` object.

**Type** `mysql.connect()`

**cursor**

A `mysql.connector.connect().cursor()` object.

**Type** `mysql.connect().cursor()`

**close\_connections** () → None

Close database connections.

Method closes database cursor and connection.

**Returns** None

**static convert\_datetime** (*dt*: <module 'datetime' from '/usr/lib/python3.6/datetime.py'>) → <module 'datetime' from '/usr/lib/python3.6/datetime.py'>

Converts python datetime to MySQL datetime.

Method converts given python datetime object to MySQL datetime format.

**Parameters** **dt** – Datetime object in default format.

**Returns** Datetime object in MySQL format.

**create\_database** () → bool

Create database

Method creates database with database name.

**Returns** True if successful, False otherwise.

**create\_table** (*tbl\_name*: str, *cols*: str) → bool

Create table in database.

Method creates table in database with given name and specifications.

**Parameters**

- **tbl\_name** – Name to give table.
- **cols** – Columns to put in table.

### Example

```
>>> tbl_name = 'table'
>>> cols = 'id INT UNSIGNED NOT NULL AUTO_INCREMENT, '
...       'permalink VARCHAR(100), message VARCHAR(100), new_date DATETIME, '
...       'userID VARCHAR(20), PRIMARY KEY(id) '
...
>>> create_table(tbl_name, cols)
```

**Returns** True if successful, False otherwise.

**delete\_message** (*tid*: str) → bool

Deletes message from message\_data table.

Method deletes given table id (row) from message\_data table.

**Parameters** **tid** – Table id from id column of message\_data table.

**Returns** True for success, False otherwise.

**describe\_table** (*tbl\_name*: str) → list

Gets description of table.

Method returns a list describing the structure of the given table.

**Parameters** **tbl\_name** – Name of table to describe

**Returns** List with table description, empty list otherwise.

**drop\_table** (*tbl\_name*: str) → bool

Drop table from database.

Function drops given table from given database.

**Parameters** `tbl_name` – Table to drop.

**Returns** True if successful, False otherwise.

**find\_comment\_id** (*cid: str*) → bool

Check comment\_list table for comment id.

Method checks comment\_list table for given comment id.

**Parameters** `cid` – Comment id to search for.

**Returns** True for success, false otherwise.

**get\_messages** (*time\_now: datetime.datetime*) → list

Retrieves list of messages that have expired.

Method returns list of messages whose expiration times are before current datetime.

**Parameters** `time_now` – Current datetime.

**Returns** List containing results of query.

**get\_rows** (*tbl\_name: str*) → tuple

Fetch table rows.

Method gets rows of given table by order of id in a tuple.

**Parameters** `tbl_name` – Name of table to get rows from.

**Returns** Tuple containing each row's data, empty tuple otherwise.

**get\_tables** () → list

Return table list of database.

Method returns a list with the names of the tables.

**Returns** List of table names.

**grant\_permissions** () → bool

Grants bot user permissions to database.

Method grants bot user permissions to database.

**Returns** True if successful, False otherwise.

## Notes

Only database root user can grant database permissions.

**lock\_read** (*tbl\_name: str*) → bool

Locks table for reading.

Method locks a given table for read access.

**Parameters** `tbl_name` – Name of table to lock.

**Returns** True if successful, False otherwise.

## Notes

- Any previous locks are [implicitly released](#).
- Read locks have lower priority than write locks.

**lock\_write** (*tbl\_name: str*) → bool

Locks table for writing.

Method locks a given table for write access.

**Parameters** **tbl\_name** – Name of table to lock.

**Returns** True if successful, False otherwise.

### Notes

- Any previous locks are [implicitly released](#).
- Write locks have higher priority than read locks.

**save\_comment\_id** (*cid: str*) → bool

Saves comment id into comment\_list table.

Method saves given comment id into the comment\_list table.

**Parameters** **cid** – Comment id to save.

**Returns** True for success, false otherwise.

**save\_message** (*link: str, msg: str, exp: <module 'datetime' from '/usr/lib/python3.6/datetime.py'>, usr\_name: str*) → bool

Saves given comment data into message\_data table.

Method saves given inputs in message\_date table as a row.

### Parameters

- **link** – Comment permalink.
- **msg** – Comment message.
- **exp** – Expiration datetime object.
- **usr\_name** – Comment author username.

**Returns** True for success, false otherwise.

**select\_database** () → bool

Select database for command execution.

Method selects database within MySQL for command execution.

**Returns** True if successful, False otherwise.

clashcallerbot\_database.**main**()

## 1.2 clashcallerbot\_reply module

Checks messages in database and sends PM if expiration time passed.

This module checks messages saved in a MySQL-compatible database and sends a reminder via PM if the expiration time has passed. If so, the message is removed from the database.

clashcallerbot\_reply.**get\_parent** (*link: str*) → str

Fetch parent comment or submission.

Function gets parent comment of given permalink or submission if top level comment.



**Parameters** **link** – Permalink to get parent of.

**Returns** Parent comment link or default string.

```
clashcallerbot_reply.main()
```

```
clashcallerbot_reply.send_reminder(link: str, msg: str, usr: str) → bool
```

Sends reminder PM to username.

Function sends given permalink and message to given username.

**Parameters**

- **link** – Permalink to comment.
- **msg** – Message in comment that was saved.
- **usr** – User to send reminder to.

**Returns** True if successful, False otherwise.

## 1.3 clashcallerbot\_search module

Searches recent reddit comments for ClashCaller! string and saves to database.

This module uses the Python Reddit API Wrapper (PRAW) to search recent reddit comments for the ClashCaller! string.

If found, the userID, permalink, comment time, message, and expiration time (if any) are parsed. The default, or provided, expiration time is applied, then all the comment data is saved to a MySQL-compatible database.

The comment is replied to, then the userID is PM'ed confirmation.

```
clashcallerbot_search.have_replied(cid: str, bot_name: str) → bool
```

Checks if bot user has replied to a comment.

Function checks reply authors for bot user.

**Parameters**

- **cid** – Comment ID to get replies of.
- **bot\_name** – Name of bot to check for.

**Returns** True if successful, False otherwise.

```
clashcallerbot_search.is_recent(cmnt: praw.models.reddit.comment.Comment) → bool
```

Checks if comment is a recent comment.

Function compares comment time with program start time.

**Parameters** **cmnt** – praw comment object.

**Returns** True if comment time is after start time, False otherwise.

```
clashcallerbot_search.main()
```

```
clashcallerbot_search.send_confirmation(u_name: str, link: str, exp: datetime.datetime) → bool
```

Send confirmation to reddit user.

Function sends given user confirmation of given expiration time with given link.

**Parameters**

- **u\_name** – username of user.

- **link** – Permalink of comment.
- **exp** – Expiration datetime of call.

**Returns** True if successful, False otherwise.

`clashcallerbot_search.send_confirmation_reply` (*cid: str, link: str, exp: datetime.datetime*)  
→ str

Replies to a comment.

Function replies to a given comment ID with a given message.

**Parameters**

- **cid** – Comment ID to reply to.
- **link** – Permalink of comment.
- **exp** – Expiration datetime of call.

**Returns** id of new comment if successful, None otherwise

`clashcallerbot_search.send_error_message` (*u\_name: str, link: str, error: str*) → bool  
Send error message to reddit user.

Function sends given error to given user.

**Parameters**

- **u\_name** – username of user.
- **link** – Permalink of comment.
- **error** – Error to send to user.

**Returns** True if successful, False otherwise.

## 1.4 logging\_conf module

Defines logging dictionary.

Module defines dictionary for logging.config.dictConfig()

## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### C

clashcallerbot\_database, 1  
clashcallerbot\_reply, 4  
clashcallerbot\_search, 5

### I

logging\_conf, 6



## INDEX

### C

clashcallerbot\_database (module), 1  
clashcallerbot\_reply (module), 4  
clashcallerbot\_search (module), 5  
ClashCallerDatabase (class in clashcallerbot\_database), 1  
close\_connections() (clashcallerbot\_database.ClashCallerDatabase method), 1  
config\_file (clashcallerbot\_database.ClashCallerDatabase attribute), 1  
convert\_datetime() (clashcallerbot\_database.ClashCallerDatabase static method), 2  
create\_database() (clashcallerbot\_database.ClashCallerDatabase method), 2  
create\_table() (clashcallerbot\_database.ClashCallerDatabase method), 2  
cursor (clashcallerbot\_database.ClashCallerDatabase attribute), 1

### D

delete\_message() (clashcallerbot\_database.ClashCallerDatabase method), 2  
describe\_table() (clashcallerbot\_database.ClashCallerDatabase method), 2  
drop\_table() (clashcallerbot\_database.ClashCallerDatabase method), 2

### F

find\_comment\_id() (clashcallerbot\_database.ClashCallerDatabase method), 3

### G

get\_messages() (clashcallerbot\_database.ClashCallerDatabase method), 3

get\_parent() (in module clashcallerbot\_reply), 4  
get\_rows() (clashcallerbot\_database.ClashCallerDatabase method), 3  
get\_tables() (clashcallerbot\_database.ClashCallerDatabase method), 3  
grant\_permissions() (clashcallerbot\_database.ClashCallerDatabase method), 3

### H

have\_replied() (in module clashcallerbot\_search), 5

### I

is\_recent() (in module clashcallerbot\_search), 5

### L

lock\_read() (clashcallerbot\_database.ClashCallerDatabase method), 3  
lock\_write() (clashcallerbot\_database.ClashCallerDatabase method), 3  
logging\_conf (module), 6

### M

main() (in module clashcallerbot\_database), 4  
main() (in module clashcallerbot\_reply), 5  
main() (in module clashcallerbot\_search), 5  
mysql\_connection (clashcallerbot\_database.ClashCallerDatabase attribute), 1

### R

root\_user (clashcallerbot\_database.ClashCallerDatabase attribute), 1

### S

save\_comment\_id() (clashcallerbot\_database.ClashCallerDatabase method), 4

`save_message()` (clashcaller-  
bot\_database.ClashCallerDatabase method),  
[4](#)

`select_database()` (clashcaller-  
bot\_database.ClashCallerDatabase method),  
[4](#)

`send_confirmation()` (in module clashcallerbot\_search), [5](#)

`send_confirmation_reply()` (in module clashcaller-  
bot\_search), [6](#)

`send_error_message()` (in module clashcallerbot\_search),  
[6](#)

`send_reminder()` (in module clashcallerbot\_reply), [5](#)