clashcallerbot-reddit Documentation

Release 2.2.0

Jose A. Lerma III

Nov 10, 2018

GETTING STARTED

ClashCallerBot was made to help /r/ClashOfClans clans to coordinate attacks during war from within reddit.

For example, someone wants to attack base 1 and 7, but they haven't posted an update in over an hour and those two bases still haven't been attacked. Is it okay to attack those bases? Did your fellow clan member die? Who knows?

Well, if s/he (or someone on their behalf) had called those bases for a set period of time, you would know for certain.

Think of **ClashCallerBot** as an independent time keeper that runs entirely within reddit.

If not obvious enough, it's a fork of SIlver–'s RemindMeBot. It's a sweet (as in awesome) little program.

I plan to make a few more tweaks to make it even more useful.

CHAPTER

ONE

USAGE INFORMATION

Usage can be inferred from the regular expressions used to process each comment in the *clashcallerbot_search module*:

```
# Regular expressions
clashcaller_re = re.compile(r'''
                            [!|\s]?             # prefix ! or space (optional)
                            [C|c]lash[C|c]aller # lower or camelcase ClashCaller (required)
                            [!|\s]              # suffix ! or space (required)
                            ''', re.VERBOSE)
expiration_re = re.compile(r'''
                           (?P<exp_digit>(\d){1,2})    # single or double digit (required)
                           (\s)?                       # space (optional)
                           (?P<exp_unit>minute(s)?\s|  # minute(s) (space after required)
                           min\s|                      # minute abbr. (space after required)
                           hour(s)?\s|                 # hour(s) (space after required)
                           hr\s                        # hour abbr. (space after required)
                           )+''', re.VERBOSE | re.IGNORECASE)  # case-insensitive
message_re = re.compile(r'''
                        (\s)*       # space (optional)
                        "           # opening double quote (required)
                        base(s)?    # string: base(s) (required)
                        [\W|\s]*    # non-word character or space (optional)
                        (\d){1,2}   # single or double digit (required)
                        .*          # any character after (optional)
                        "           # closing double quote (required)
                        ''', re.VERBOSE | re.IGNORECASE)  # case-insensitive
```

To sum:

- The `clashcaller` string must be present in either lower or camelcase with an exclamation point `!` either before or after.
- The expiration time in minutes or hours must follow either abbreviated or with full spelling with an optional space between the number and word, but mandatory space after the word: `4min`. Case is ignored.
- The message within quotes must follow with the singular or plural form of `base` and a required single or double digit number. Case is ignored.

1.1 Quickstart

This is what you'll need to run **clashcallerbot-reddit** ASAP (As Soon As Possible). **clashcallerbot-reddit** is developed on various Linux distros, so *Installation* is assumed to be on Linux.

1.1.1 Prerequisites

Python **clashcallerbot-reddit** is a massive *python* script, so at least Python 3.6 is needed.

Pip Configuring *python* is easiest using pip. Once pip is installed, the requirements can be installed by running `pip install -r requirements.txt` from within the source code directory.

MySQL **clashcallerbot-reddit** uses a MySQL-compatible database to store calls and to keep track of comments that were replied to. Either MySQL or MariaDB can be used.

With these prerequisites met, **clashcallerbot-reddit** can be setup and run.

1.1.2 Setup

First, add the bot's reddit metadata to *praw-example.ini* and rename to *praw.ini*, then add the database's root and desired bot user credentials to *database-example.ini* and rename to *database.ini*.

Next, change the following line in *clashcallerbot_database module*

```
462     # Create the clashcaller database
463     database = ClashCallerDatabase(config_file=config, root_user=False)
```

to `database = ClashCallerDatabase(config_file=config, root_user=True)`. This may get updated to be default later.

Now, the MySQL-compatible database can be setup by running clashcallerbot_database.py directly from within terminal:

```
python3 ./clashcallerbot_database.py
```

1.1.3 Starting

Once the database is setup, the bot can be run by calling *clashcallerbot_search module* and *clashcallerbot_reply module* directly from within terminal:

```
python3 ./clashcallerbot_search.py && python3 ./clashcallerbot_reply.py
```

Alternatively, by running the provided bash script from within terminal:

```
./clashcallerbot.sh
```

Note:

- Remember to set executable mode with `chmod +x ./clashcallerbot.sh`.
- Script assumes files are in same directory and is run as crontab in *No Root Setup*.
- How often to run as a crontab depends on how long you want the bot to be down/broken.
- If you have access to root, check *Installation* for info on setting up systemd instead.

- Logfile can be removed if not necessary (remove variable and `>> $logfile`).
- `python2.7` can be replaced with relevant python version.
- The function in `kill` returns all script PIDs, so it must be restarted.
- If you have access to `pidof`, you can avoid killing all script instances.

1.1.4 Updating

A bash script is also provided for updating and can be run directly from within terminal:

```
./redownload.sh
```

Note:

- Don't forget to set executable mode with `chmod +x ./redownload.sh`.
- Script assumes files are all in same directory and that it is run in a *No Root Setup*.
- The `-f` switch and `> /dev/null 2>&1` silence outputs to certain degrees.

1.2 Installation

This thing has to run 24/7. As a proof of concept, I set it up in an Ubuntu VM for the open alpha and a CentOS VM for the open beta. Production version was running on CloudLinux OS at one point, but has since moved to Amazon Linux AMI on AWS. If you are using a different OS, your mileage may vary.

1.2.1 Deployment Options

Believe me, the list of where I have not deployed it is shorter. I mostly went with virtual machines for the development, and tried putting the production version on a shared host. It has been migrated to an Amazon EC2 instance in hopes of minimizing costs.

Amazon Linux AMI Setup

Amazon Linux AMI is a fork of RHEL6 (Red Hat Enterprise Linux version 6) made to run on the EC2 (Amazon Elastic Compute Cloud) of AWS (Amazon Web Services). It is a good choice for having an operating system that is fully up to date and supported by the "host." I particularly like that it is already setup with most of what is needed. If preferred, use Ubuntu, OpenSUSE, Red Hat, et al., but your mileage will vary using this guide.

Tip: Check out the *CentOS Setup Guide* or the *Ubuntu Setup Guide* if using those distros.

Get an AWS account set up/prep for using EC2, then set up the EC2 instance itself. It is a bit of an orchestration, but worth it.

Note:

- Opted for a t2.micro instance since the free tier lasts 12 months, though that may change to a t1.micro. Also used Security Groups to limit access to My IP and used a key pair.
- It is a good idea to enable billing alerts.

Configure an EBS (Elastic Block Store) to store the database. Went with 22 GB since it is free, but I may change that to around 10 GB later. Also went with magnetic storage because it costs less than SSD storage and the I/O speed is not needed right now.

Note:

- The root volume is deleted on termination of the instance, so I turned on termination protection. Alternatively, the root volume can be changed to persist.
- When connecting from Windows operating systems, I prefer PuTTY/KiTTY, but there is a doc detailing setup.

Next, secure the EC2 instance. It is an old guide, so the only thing that needs to be done is adding a new user (with limited access, by default) to run the bot. At this point, a root volume snapshot can be made to save progress, if persistence was not enabled previously.

Tip:

- Install and enable yum-cron to keep the EC2 instance updated automatically.

Install some dependencies as the **default user**, not the new one.

```
sudo yum install gcc python-devel openssl-devel libffi-devel
pip install --upgrade pip
```

As the **new user**, set up the needed environment. From within the virtual environment, run:

```
pip install -U wheel
pip install praw
```

Set up a MySQL database within an EBS volume as the **default user**. The guide is for Ubuntu, but setup for Amazon Linux is very similar. Once the EBS volume is created and attached, the **default user** needs to run the following from within the Amazon Linux EC2 instance to create an XFS filesystem at `/vol`:

```
# Create XFS filesystem
sudo yum install xfsprogs mysql-server mysql-devel
grep -q xfs /proc/filesystems || sudo modprobe xfs
sudo mkfs.xfs /dev/sdb # change to wherever volume is mounted

# Mount XFS filesystem
echo "/dev/sdb /vol xfs noatime 0 0" | sudo tee -a /etc/fstab
sudo mkdir -m 000 /vol
sudo mount /vol
```

Now that MySQL is installed, it must be configured.

```
sudo service mysqld start
sudo service mysqld status    # Confirm it is running
sudo mysql_secure_installation    # Say 'y' to everything!
sudo mysql -uroot -p"password"
```

From within the MySQL prompt, `mysql>`, the database can be set up.

```
CREATE DATABASE db_name;
USE db_name;
CREATE TABLE message_table (id INT UNSIGNED NOT NULL AUTO_INCREMENT, permalink␣
↪VARCHAR(100), message VARCHAR(100),
new_date DATETIME, userID VARCHAR(20), PRIMARY KEY(id));
ALTER TABLE message_table AUTO_INCREMENT=1;
CREATE TABLE comment_table (id MEDIUMINT NOT NULL, list VARCHAR(35), PRIMARY KEY(id));
INSERT INTO comment_table VALUES (1, "'0'");
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, INDEX, ALTER ON db_name.* TO 'botname
↪'@localhost IDENTIFIED BY
'password';
QUIT
```

Make sure that MySQL is stopped with `sudo service mysqld stop && sudo service mysqld status`, then move MySQL into the EBS volume.

```
sudo mkdir /vol/etc /vol/lib /vol/log
sudo mv /etc/my.cnf /vol/etc/
sudo mv /var/lib/mysql /vol/lib/
sudo mv /var/log/mysqld.log /vol/log

sudo ln -s /vol/etc/my.cnf /etc/my.cnf
sudo ln -s /vol/log/mysqld.log /var/log/mysqld.log

sudo mkdir /var/lib/mysql
echo "/vol/lib/mysql /var/lib/mysql none bind" | sudo tee -a /etc/fstab
sudo mount /var/lib/mysql

sudo service mysqld start && sudo service mysqld status
```

Now that the database has been set up, more dependencies need to be installed in the virtual environment as the **new user**.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
pip install mysql-connector
```

Once all relevant files have been added, the bot can be started, output redirected to a null terminal, and process put in the background.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
nohup python3 clashcallerbot_reply.py > /dev/null 2>&1 &
nohup python3 clashcallerbot_search.py > /dev/null 2>&1 &
```

Tip:

- The bot has to login to reddit at least once to refresh the oauth token. Amazon Linux does not have a web browser installed by default, so run `sudo yum install lynx` as the **default user** before running the script.

CentOS 7.0 Setup

How you want CentOS installed is your choice. I went with the xxx-Minimal.iso so that it would be as lean as possible (since I planned for it to be in the cloud) and would only be as large as necessary to run the bot. Installation settings are also up to you (security, networking, user accounts, etc.). Since it is a minimal install, there is no GUI (Graphical User Interface) and everything is run from TTY1 (TeleTYpewriter number 1).

First, install EPEL (Extra Packages for Enterprise Linux). Pip isn't even available, so we start here:

```
yum install epel-release
```

Enable compiling from source. The list of what won't be compiled from source is shorter.

```
yum install gcc python-devel openssl-devel libffi-devel
```

Install and upgrade pip and wheel,:

```
yum install python-pip python-wheel && pip install -U pip wheel
```

Then, set up the needed environment. From within the virtual environment, run:

```
pip install praw
```

Now, setup MySQL/MariaDB.

```
yum install mariadb-server mariadb mariadb-devel
systemctl start mariadb     # start mariadb
systemctl enable mariadb     # start at boot
systemctl status mariadb    # confirm installation
mysql_secure_installation # Say yes to everything!
mysql -uroot -p"password" # login to mysql
```

From within the MySQL prompt, setup the database itself.

```
CREATE DATABASE db_name;
USE db_name;
CREATE TABLE message_table (id INT UNSIGNED NOT NULL AUTO_INCREMENT, permalink␣
↪VARCHAR(100), message VARCHAR(100),
new_date DATETIME, userID VARCHAR(20), PRIMARY KEY(id));
ALTER TABLE message_table AUTO_INCREMENT=1;
CREATE TABLE comment_table (id MEDIUMINT NOT NULL, list VARCHAR(35), PRIMARY KEY(id));
INSERT INTO comment_table VALUES (1, "'0'");
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY␣
↪TABLES, LOCK TABLES ON db_name.*
TO 'botname'@localhost IDENTIFIED BY 'password';
```

Now that MySQL is setup, install more dependencies.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
pip install mysql-connector
```

Start, redirect output, and background process.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
nohup python clashcallerbot_reply.py > /dev/null 2>&1 &
nohup python clashcallerbot_search.py > /dev/null 2>&1 &
```

Tip: Alternatively, use systemd service.

No Root Setup

What system you install to is your choice, but this setup assumes a CloudLinux OS without root access and with both Python and MySQL installed. Everything is run from terminal (ssh compatible).

Note: If you have multiple Python versions, replace `python` and `pip` with your version, e.g., `python2.7 get-pip.py --user`, `pip2.7 install --user -U pip wheel`.

Since root is not available, pip needs to be installed locally.

```
wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py --user
```

To `~/.bashrc`, add `PATH=$PATH:~/.local/bin` and `PYTHONPATH=$PYTHONPATH:~/.local/lib/python/site-packages/`, then run `source ~./bashrc` to apply changes.

Next, update pip and wheel,:

```
pip install --user -U pip wheel
```

Then, set up the needed environment. From within the virtual environment, run:

```
pip install --user praw
```

From within the MySQL prompt, setup the database itself.

```
CREATE DATABASE db_name;
USE db_name;
CREATE TABLE message_table (id INT UNSIGNED NOT NULL AUTO_INCREMENT, permalink␣
↪VARCHAR(100), message VARCHAR(100),
new_date DATETIME, userID VARCHAR(20), PRIMARY KEY(id));
ALTER TABLE message_table AUTO_INCREMENT=1;
CREATE TABLE comment_table (id MEDIUMINT NOT NULL, list VARCHAR(35), PRIMARY KEY(id));
INSERT INTO comment_table VALUES (1, "'0'");
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY␣
↪TABLES, LOCK TABLES ON db_name.*
TO 'botname'@localhost IDENTIFIED BY 'password';
```

Now that MySQL is setup, install more dependencies.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
pip install --user mysql-connector
```

Start, redirect output, and background process.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
nohup python clashcallerbot_reply.py > /dev/null 2>&1 &
nohup python clashcallerbot_search.py > /dev/null 2>&1 &
```

Tip: Alternatively, use systemd service.

Ubuntu 14.04 LTS Setup

Ubuntu Desktop is a good choice because most IDEs need a GUI, but other versions are available (like Server). The following instructions are a combination of setting up the Ubuntu OS to run the bot (build packages and setting up MySQL tables) as well as adding all the needed dependencies into python. Everything here is run in a terminal.

First, enable compiling from source. There will be a lot of compiling from source in the future...

```
sudo apt-get install build-essential python-dev libffi-dev libssl-dev
```

Next, set up pip:

```
sudo apt-get install python-pip && sudo pip install pip
```

Then, set up the needed environment. From within the virtual environment, run:

```
sudo pip install praw
```

Now, setup MySQL/MariaDB.

```
sudo apt-get install mysql-server # For MariaDB: sudo apt-get install mariadb-server␣
↪mariadb
                                  # Set MySQL `root` pw when prompted (recommended)
mysql_secure_installation         # say yes to everything after first prompt
sudo apt-get install libmysqlclient-dev # For MariaDB: sudo apt-get install␣
↪libmariadbclient-dev
```

From within the MySQL/MariaDB prompt, set up the database itself.

```
mysql -uroot -p"password" # login as root
CREATE DATABASE db_name;
USE db_name;
CREATE TABLE message_table (id INT UNSIGNED NOT NULL AUTO_INCREMENT, permalink␣
↪VARCHAR(100), message VARCHAR(100),
new_date DATETIME, userID VARCHAR(20), PRIMARY KEY(id));
ALTER TABLE message_table AUTO_INCREMENT=1;
CREATE TABLE comment_table (id MEDIUMINT NOT NULL, list VARCHAR(35), PRIMARY KEY(id));
INSERT INTO comment_table VALUES (1, "'0'");
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY␣
↪TABLES, LOCK TABLES ON db_name.*
TO 'botname'@localhost IDENTIFIED BY 'password';
```

Now that MySQL/MariaDB is set up, install more dependencies.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
sudo pip install mysql-connector
```

Start, redirect output, and background process.

```
source clashcallerbot-reddit/bin/activate    # set virtual environment, if needed
nohup python3 clashcallerbot_reply.py > /dev/null 2>&1 &
nohup python3 clashcallerbot_search.py > /dev/null 2>&1 &
```

Ubuntu

For the open alpha, I went with an OS that is convenient (need something? There's a package for that) as a proof of concept and debugging environment.

Ubuntu 14.04 LTS Setup

CentOS

For the open beta, I went with an OS that is often available for production environments.

CentOS 7.0 Setup

No Root

For my first attempt at a production version, I got a shared host running CloudLinux OS that does not provide root access, but has both Python and MySQL pre-installed.

No Root Setup

Amazon EC2

For what I hope is the final production version, I set up an Amazon EC2 instance running Amazon Linux AMI. I particularly like that it has pip and python pre-installed.

Amazon Linux AMI Setup

1.2.2 Building Documentation

Note: Building the documentation is **not needed or recommended** unless contributing to the documentation. The latest version of the documentation is available at josealermaiii@github.io/clashcallerbot-reddit or as a PDF in the source code. You have been warned.

Building the docs requires a few more pip packages:

- Sphinx
- sphinxcontrib-napoleon
- sphinx-rtd-theme

If that wasn't bad enough, a symbolic link to *praw.ini* must be made from within `~/.config`.:

```
cd ~/.config
ln -s absolute_path_here/clashcallerbot-reddit/praw.ini praw.ini
```

This has to do with how *praw* finds the *praw.ini* file. This may get fixed later much like the *database.ini* fix in *config.py*.

Now, we can build the docs in HTML format:

```
cd absolute_path_here/clashcallerbot-reddit/docs
make html
```

This will save the docs website in `../../clashcallerbot-reddit-docs/`.

Building the PDF is even more involved. First, LaTeX must be installed on the OS. For example, in Ubuntu 18.04:

```
sudo apt-get install texlive-latex-recommended texlive-latex-extra texlive-fonts-
↪recommended
```

Installing these dependencies is not recommended, if not needed, because they require > 300 MB of disk space.

Now, we can build the docs in PDF format:

```
cd absolute_path_here/clashcallerbot-reddit/docs
make latexpdf
```

This will save the doc's PDF in `../manual.pdf`.

1.2.3 Disclaimer

Though covered by the MIT License, I reiterate: executing code from the Internet in terminal can end up doing bad things.

> Read and understand all code you copy and paste before running it.

1.3 clashcallerbot-reddit

1.3.1 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() → bool
Close database connections.

Method closes database cursor and connection.

Returns True if successful, False otherwise.

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.

open_connections() → bool
Open database connections.

Method makes database connection and cursor.

Returns True if successful, False otherwise.

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**()

1.3.2 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.

1.3.3 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.

1.3.4 logging_conf module

Defines logging dictionary.

Module defines dictionary for logging.config.dictConfig()

logging_conf.**LOGGING**

Dictionary containing definitions for the loggers, handlers, and formatters.

Type dict

1.4 References

clashcallerbot-reddit wasn't built with inherent knowledge. These are all the reference materials used to help make the bot run properly. Truth be told, these are all the bookmarks in my `clashcallerbot` folder.

1.4.1 Python

- logging.config - Logging configuration
- datetime - Basic date and time types
  - strftime() and strptime() Behavior
- re - Regular expression operations
- configparser - Configuration file parser

1.4.2 PRAW

- Submission Stream Reply Bot
- Comment Extraction and Parsing
- SubredditStream
- Comment
- Redditor
- praw/config.py at v6.0.0

1.4.3 Database

- MySQL 8.0 Reference Manual
  - Internal Locking Methods
- How to connect Python programs to MariaDB
- MySQL Connector/Python Developer Guide - Connector/Python Coding Examples
- MySQL Connector/Python Developer Guide - Guidelines for Python Developers

1.4.4 Documentation

- Setting up Sphinx for generating documentation from DocStrings, leveraging the Napoleon extension
  - Sphinx for Python documentation - Giselle Zeno
    - Publishing sphinx-generated docs on github - sphinxdoc-test v0.1 documentation
  - Example Google Style Python Docstrings - napoleon 0.7 documentation
  - reStructuredText Primer - Sphinx 2.0.0+ documentation
- Semantic Versioning 2.0.0
- LaTeX error attempting to build PDF on Ubuntu 11.04

1.4.5 Miscellaneous

- JoseALermaIII/clashcallerbot-reddit
- Regex Tester
- Testing Thread Trois: ClashCallerBot

- commenting - reddit.com
- Logging module is causing Sphinx to crash. - Google Groups

1.5 Index

PYTHON MODULE INDEX

C

L

INDEX

S