Contents

1 Getting Started 3
  1.1 Overview ................................................. 3
  1.2 Dashboard .............................................. 4
  1.3 Admin Panel ............................................ 5
  1.4 Architecture ......................................... 6
  1.5 Features .............................................. 8
  1.6 Utility ................................................ 8

2 Installation 9
  2.1 Overview ................................................. 9
  2.2 Installation with Asterisk ................................. 10
  2.3 Installing on FreePBX .................................. 11
  2.4 Installation with FreeSWITCH ............................ 11
  2.5 Broker Installation .................................... 12
  2.6 Celery Installation .................................... 14

3 Configuration and Defaults 15
  3.1 General Configuration .................................. 15
  3.2 Country Reporting ...................................... 17
  3.3 Configuration for Asterisk ............................... 18
  3.4 Realtime configuration for Asterisk ..................... 19
  3.5 Configuration for FreeSWITCH ........................... 19
  3.6 Realtime configuration for FreeSWITCH .................. 19
  3.7 Resetting CDR Data .................................... 20
  3.8 Celery Configuration .................................... 21
  3.9 ACL Control ........................................... 23

4 Troubleshooting 25
  4.1 Where to find the log files .............................. 25
  4.2 Run in debug mode ..................................... 25
  4.3 Celerymon ............................................. 25

5 User Guide 27
  5.1 Overview .............................................. 27
  5.2 How to use CDR-Stats .................................. 28
  5.3 Admin Panel ........................................... 28
  5.4 Customer Panel ....................................... 33

6 PostgreSQL 49
CDR-Stats is free and open source call detail record analysis and reporting software for Freeswitch, Asterisk and other type of VoIP Switch. It allows you to interrogate your CDR to provide reports and statistics via a simple to use, yet powerful, web interface.

It is based on the Django Python Framework, Celery, Gevent, PostgreSQL and InfluxDB.

1.1 Overview

CDR-Stats is an application that allows browsing and analysing CDR (Call Detail Records).

Different reporting tools are provided:

- Dashboard: Overview of call activity
- Search CDR: Search, filter, display and export CDR
- Overview: Analyse call traffic by hour, day and month
- Daily Comparison: Compare call traffic day on day
- Real-Time Statistics: Show concurrent calls in realtime by switch
- Concurrent Calls: Concurrent Calls through the day updated in real-time
- Country Report: Call statistics by country
- World Map: Call statistics overlaid on a world map
• Mail daily aggregated reports
• Threat Control: Detect abnormal call patterns
• Destination Alerts: Unexpected destination alerts

CDR Stats uses PostgreSQL, a scalable, high performance database system used to analyze large quantities of CDR data. PostgreSQL provides Materialized views which make it perfect to build analytic applications which do heavy aggregation and recently PostgreSQL came with Jsonb field which make it easy to store custom data from variety of switch.

Out of the box, CDR-Stats supports Freeswitch and Asterisk using connectors that get the CDR. Connectors for other switch systems can be built.

Other Switch support has been added, please refer to http://www.cdr-stats.org/connectors

1.2 Dashboard

User Dashboard provides real-time monitoring of the most relevant metrics of connected switches.
1.3 Admin Panel

The Admin Panel allows the administrators to configure the entire reporting platform, import CDR in csv format, configure users, switch connections and automatic alarms.
1.4 Architecture

CDR-Stats uses PostgreSQL as the underlying CDR store. PostgreSQL with Materialized view allows querying and analysis of many millions of records without noticeable loss of performance, and can easily be scaled as demand increases.

Postgresql is used for managing CDR-Stats in terms of users and managing the web framework, Django.

Celery, a task manager runs in the background, and monitors the CDR coming into the system, and alerts the systems administrator when unusual behaviour is discovered. What is determined as unusual behaviour is determined by the administrator who can configure alerts for increases in dropped calls, average length of calls, or calls to unusual destinations.

At the moment Freeswitch and Asterisk are supported, for other switches such as OpenSIPs or Kamailio, connectors can be built to connect to the CDR database store and import them in realtime to CDR-Stats.
1.4. Architecture
1.5 Features

Many features are provided on CDR-Stats, from browsing millions of CDRs, providing efficient search facilities to build reporting such as monthly reports, concurrent calls view, and comparing call traffic with previous days.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telephony Reporting</td>
<td>Leading open source switches Freeswitch, Asterisk, supported as standard.</td>
</tr>
<tr>
<td>Multi-switch</td>
<td>Monitor traffic from many switches in one location</td>
</tr>
<tr>
<td>Multi-tenant</td>
<td>Allowing many customers to monitor their own CDR on one instance of CDR-Stats.</td>
</tr>
<tr>
<td>Distributed</td>
<td>Runs on one or more machines. Supports broker clustering and HA. New workers can be set up without central configuration.</td>
</tr>
<tr>
<td>Fraud detection</td>
<td>Visualise traffic which helps to identify unusual patterns.</td>
</tr>
<tr>
<td>Fraud Alert</td>
<td>Send emails to the administrator when fraud are or suspicious patterns occur</td>
</tr>
<tr>
<td>Error Emails</td>
<td>Can be configured to send emails to the administrator if a task fails.</td>
</tr>
<tr>
<td>Import CDR</td>
<td>Import CDR files in custom format</td>
</tr>
<tr>
<td>World Map view</td>
<td>See where the traffic originates and terminates on a Map</td>
</tr>
<tr>
<td>Compare traffic</td>
<td>See how your traffic evolves, and patterns change.</td>
</tr>
<tr>
<td>Mail Reporting</td>
<td>Send daily mail reports of telecoms traffic</td>
</tr>
<tr>
<td>Realtime Reporting</td>
<td>Traffic displayed in realtime</td>
</tr>
<tr>
<td>Blacklist</td>
<td>Blacklist Phone number patterns to receive alarms</td>
</tr>
<tr>
<td>Geographic alerts</td>
<td>Set alert if calls go to disallowed countries</td>
</tr>
<tr>
<td>Concurrent calls</td>
<td>Realtime reporting of concurrent calls</td>
</tr>
</tbody>
</table>

1.6 Utility

CDR-Stats is a simple-to-use tool to provide easy analysis of calls. It is a recommended addition to telephony servers, whether it be a simple in-house PBX or large capacity VoIP switch. It shows in near realtime what calls are going through, can detect errors and failures, and alert the systems administrator if unexpected traffic is noted.
CHAPTER 2

Installation

Contents:

2.1 Overview

2.1.1 Install requirements

A Requirements file provides a way to create an environment where all the optional dependencies needed for the Project/Application are installed.

To get started with CDR-Stats the following must be installed:

- python >= 2.5 (programming language)
- Apache / http server with WSGI modules
- Django Framework >= 1.4 (Python based Web framework)
- Celery >= 3.0 (Asynchronous task queue/job queue based on distributed message passing)
- django-celery >= 3.0 (Celery integration for Django)
- linaro_django_pagination (Utilities for creating robust pagination tools throughout a django application)
- django-uuidfield >= 0.2 (Provides a UUIDField for your Django models)
- django-reusableapps >= 0.1.1 (Python module to enable Django to load reusable, pluggable and egg-based applications)
- docutils >= 0.7 (Text processing system for processing plaintext documentation into useful formats)
- kombu >= 1.0.2 (An AMQP - Advanced Message Queuing Protocol messaging framework for Python)
- pyparsing >= 1.5.5 (A general parsing module for Python)
- python-dateutil >= 1.5 (Extensions to the standard datetime module)
- redis >= 2.2.2 (Redis Python Client)
- uuid >= 1.30 (UUID object and generation functions)
- wsgiref >= 0.1.2 (Validation support for WSGI)
- django-notification >= 0.1.3 (User notification management for the Django web framework)
- switch2bill-common - Common libs reused in different projects
- django-country-dialcode - Django reusable application to manage Dial code of Countries
• django-countries - List of world countries

The requirements are installed into a virtual environment so that the dependencies of the application do not interfere with other applications on the server. More information can be found about virtualenv at: http://pypi.python.org/pypi/virtualenv


With PIP you can easily install all the requirements:

$ pip install -r install/requirements/all-requirements.txt

2.1.2 Running CDR-Stats

Inside CDR-Stats directory you should run, the following:

$ python manage.py syncdb --noinput
$ python manage.py collectstatic
$ python manage.py migrate
$ python manage.py createsuperuser
$ python manage.py runserver

syncdb will create a database named test.db in database folder of the CDR-Stats directory. CDR-Stats is configured to do this, but can be changed by modifying settings.py where the DATABASES dictionary is constructed. there is more information about this in the Django documentation.

collectstatic will fetch all necessary media files and put them into static folder defined in the settings module.

migrate will applying database migration to update the database schemas of CDR-Stats to its latest version.

createsuperuser will create a super user, to access to the admin section of CDR-Stats.

runserver runs an embedded webservice to test the site. By default it will run on http://localhost:8000. This is configurable and more information about runserver is in Django documentation.

2.2 Installation with Asterisk

2.2.1 Installation via Script

Before commencing installation, it is necessary that Asterisk is configured to write CDR to a MySQL database. If this has not been done already, there are some resources to configure Asterisk to write its CDR records to MySQL at http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/asterisk-SysAdmin-SECT-1.html

It is wise to take a backup of the CDR database. A note needs to be taken of the CDR database name, the CDR table, as well as the MySQL root password as this will be required during the installation of CDR-Stats.

Run the following commands at the console:

$ bash ./install-cdr-stats-asterisk.sh
The install routine will ask a number of questions, all of which are self explanatory.

Note that CDR-Stats can be installed on the same server as Asterisk, or on a separate server connecting remotely to the Asterisk CDR database.

### 2.3 Installing on FreePBX

CDR-Stats will be configured to attach to the asteriskcdrdb database in MySQL installed by the FreePBX installation routine, which contains all the call data records. A connector is installed that takes the CDR from MySQL and imports them into PostgreSQL in realtime. The web interface for CDR-Stats will be installed on port 8008 and the Websocket on port 9000, so it is necessary to update the firewall settings to allow access to these ports.

Before commencing, a back up of FreePBX, in particular asteriskcdrdb is recommended. Also have a note of the root password for MySQL.

Run the following commands at the console:

```
$ bash install-cdr-stats-asterisk.sh
```

The install routine will ask a number of questions, all of which are self explanatory. Select “Install all” which is option 1 in the CDR-Stats Installation Menu.

### 2.4 Installation with FreeSWITCH

#### 2.4.1 Installation via Script

On an existing installation of Freeswitch, mod_cdr_sqlite needs to be compiled into Freeswitch. This procedure is described at [https://wiki.freeswitch.org/wiki/Mod_cdr_sqlite](https://wiki.freeswitch.org/wiki/Mod_cdr_sqlite)

After having recompiled Freeswitch to support Sqlite CDR, make the following changes:

In freeswitch/conf/autoload_configs/cdr_sqlite.conf.xml

Change:

```xml
<param name="legs" value="a"/>
```

To:

```xml
<param name="legs" value="ab"/>
```

Then reload the Freeswitch configuration.

**Now run the following commands at the console:**

```
$ bash install-cdr-stats.sh
```

When prompted, chose the option to install the Freeswitch version.

The install routine will ask a number of questions, all of which are self explanatory.
2.4.2 Installation on New Server

Another script is available to install Freeswitch along with CDR-Stats. This script is intended to be run on a fresh Debian 7.X or CentOS 6.X installation:

```bash
$ bash install-all-cdr-stats-freeswitch.sh
```

The install routine will ask a number of questions, all of which are self explanatory.

2.5 Broker Installation

This document describes the installation of two different Brokers. One is Redis and second is Rabbitmq. You can install either to work with CDR-Stats, although CDR-Stats automated install script installs Redis.

2.5.1 Redis

**Download Source**

Download: redis-server_2.0.0~rc2-1_amd64.deb.

**To install Redis-Server**

```bash
$ sudo dpkg -i redis-server_2.0.0~rc2-1_amd64.deb
```

or you can use apt-get

```bash
$ apt-get install redis-server
```

**Running Server**

```bash
$ redis-server
```

2.5.2 Rabbitmq

RabbitMQ is a complex and sophisticated product. If you don’t need this level of robustness, then you might want to take a look at Redis - it installs easily, runs relatively lean, and can be monitored and maintained without a lot of fuss.

See Installing RabbitMQ over at RabbitMQ’s website.

**Note:** If you’re getting `nodedown` errors after installing and using `rabbitmqctl` then this blog post can help you identify the source of the problem:

http://somic.org/2009/02/19/on-rabbitmqctl-and-badrpcnodedown/

**Download Source**

http://www.rabbitmq.com/server.html
Debian APT repository

To make use of the RabbitMQ APT repository,

1. Add the following line to your /etc/apt/sources.list

   ```
   deb http://www.rabbitmq.com/debian/ testing main
   ```

   **Note:** The word testing in the above line refers to the state of the release of RabbitMQ, not any particular Debian distribution. You can use it with Debian stable, testing or unstable, as well as with Ubuntu. In the future there will be a stable release of RabbitMQ in the repository.

2. (optional) To avoid warnings about unsigned packages, add RabbitMQ’s public key to your trusted key list using apt-key(8)

   ```
   $ wget http://www.rabbitmq.com/rabbitmq-signing-key-public.asc
   $ sudo apt-key add rabbitmq-signing-key-public.asc
   ```

3. Run apt-get update.

4. Install packages as usual; for instance,

   ```
   $ sudo apt-get install rabbitmq-server
   ```

Setting up RabbitMQ

To use celery we need to create a RabbitMQ user, a virtual host and allow that user access to that virtual host:

```
$ rabbitmqctl add_user myuser mypassword
$ rabbitmqctl add_vhost myvhost
$ rabbitmqctl set_permissions -p myvhost myuser ".*" ".*" ".*"
```

See the RabbitMQ Admin Guide for more information about access control.

Starting/Stopping the RabbitMQ server

To start the server:

```
$ sudo rabbitmq-server
```

you can also run it in the background by adding the -detached option (note: only one dash):

```
$ sudo rabbitmq-server -detached
```

Never use **kill** to stop the RabbitMQ server, but rather use the **rabbitmqctl** command:

```
$ sudo rabbitmqctl stop
```

When the server is running, continue reading Setting up RabbitMQ.
2.6 Celery Installation

2.6.1 Celery

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.

You can install Celery either via the Python Package Index (PyPI) or from source:

```
$ pip install celery
```

**Downloading and installing from source**

To Download the latest version click here.

You can install it by doing the following:

```
$ tar xvfz celery-X.X.X.tar.gz
$ cd celery-X.X.X
$ python setup.py build
$ python setup.py install # as root
```

**Using the development version**

You can clone the repository by doing the following:

```
$ git clone git://github.com/ask/celery.git
```
CHAPTER 3

Configuration and Defaults

Contents:

3.1 General Configuration

Some of the more important parts of the configuration module for the cdr_stats, settings_local.py, are explained below.

APPLICATION_DIR now contains the full path of the project folder and can be used elsewhere in the settings.py module so that the project may be moved around the system without having to worry about changing any hard-coded paths.

```
import os.path
APPLICATION_DIR = os.path.dirname(globals()['__file__'])
```

Turns on debug mode allowing the browser user to see project settings and temporary variables.

```
DEBUG = True
```

Sends all errors from the production server to the admin’s email address.

```
ADMINS = ( ('xyz', 'xyz@abc.com') )
```

Sets up the options required for Django to connect to your database engine:

```
DATABASES = {
    'default': {
        # Add 'postgresql_psycopg2', 'postgresql', 'mysql', 'sqlite3', 'oracle'
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'DATABASENAME',
        'USER': 'DB_USERNAME',
        'PASSWORD': 'DB_PASSWORD',
        'HOST': 'DB_HOSTNAME',
        'PORT': 'DB_PORT',
        'OPTIONS': {
            #Postgresql Autocommit
            'autocommit': True,
        }
    },
    'import_cdr': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'cdr-pusher',
        'USER': 'postgres',
    }
}
```
As it can be noted, there is 2 database connections, ‘default’ is the main database of CDR-Stats this is where all the table lives. The second database ‘import_cdr’ is used to import the CDRs from your switch, this database could live on an other database server but ideally it can live very happily on your CDR-Stats box. CDR-Stats doesn’t pull CDRs from your switch, it’s the job of your switch to push the CDRs to CDR-Stats.

You will need a mechanism in place to get your CDRs to the ‘import_cdr’ database, to help on this we created CDR-pusher project. CDR-Pusher will be installed on your switch server, CDR-Pusher is a Go project that can be extended, it could import CDRs from a different CDRs Database (SQLite, PostgreSQL) and/or from CDR logs files. More info please visit https://github.com/areski/cdr-pusher

Tells Django where to find your media files such as images that the HTML templates might use.

```
MEDIA_ROOT = os.path.join(APPLICATION_DIR, 'static')
```

```
ROOT_URLCONF = 'urls'
```

Tells Django to start finding URL matches at in the urls.py module in the cdr_stats project folder.

```
TEMPLATE_DIRS = ( os.path.join(APPLICATION_DIR, 'templates'), )
```

Tells Django where to find the HTML template files:

```
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
    'django.contrib.admin',  
    ...  
    'cdr',  
    'cdr_alert',  
    ...
)
```

Tells Django which applications (custom and external) to use in the project. The custom applications, cdr etc. are stored in the project folder along with these custom applications.

### 3.1.1 Mail server

To configure the SMTP client so that reports and alerts are sent via email, edit /usr/share/cdrstats/settings_local.py, and identify the email section:

```
#EMAIL_BACKEND
#===============
# Email configuration
DEFAULT_FROM_EMAIL = 'CDR-Stats <cdr-...@localhost.com>'
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
```
EMAIL_PORT = 587
EMAIL_HOST_USER = 'user...@gmail.com'
EMAIL_HOST_PASSWORD = 'password'
EMAIL_SUBJECT_PREFIX = '[CDR-Stats] '

Fill in the details to match your SMTP server. The above example is for Gmail. When done, restart Celery and Apache.

To test that the email is working, from the command line type:

```
$ cd /usr/src/cdr-stats/
$ workon cdr-stats
$ python manage.py send_daily_report
```

### 3.2 Country Reporting

CDR-Stats is able to identify the destination country of the call. This is a useful fraud prevention measure, so that calls to unexpected destinations are immediately apparent. Places that should not be called should be added in the Blacklist in the admin section so that these destinations are highlighted in the call data records.

However, in order to get accurate reporting, the call detail records have to be in international format, e.g. in the USA, this means 11 digit numbers, beginning with a 1, and for other countries, the numbers called should be prefixed with the international dial code.

There is a facility for manipulating the dialled digits reported in the call detail records, as well as identifying calls as internal calls. This is done in the “general” section of /usr/share/cdrstats/settings_local.py.

#### 3.2.1 1. Prefix Limits

PREFIX_LIMIT_MIN & PREFIX_LIMIT_MAX are used to determine how many digits are used to match against the dialcode prefix database, e.g:

```
PREFIX_LIMIT_MIN = 2
PREFIX_LIMIT_MAX = 5
```

#### 3.2.2 2. Phone Number Length

If a phone number has less significant digits than PN_MIN_DIGITS it will be considered an extension:

```
PN_MIN_DIGITS = 6
PN_MAX_DIGITS = 9
```

*NB The Number of significant digits does not include national (0) or international dialing codes (00 or 011), or where 9 is pressed for an outside line."

#### 3.2.3 3. Adding Country Code

If a phone number has more digits than PN_DIGITS_MIN but less than PN_DIGITS_MAX then the phone number will be considered as local or national call and the LOCAL_DIALCODE will be added:

```
LOCAL_DIALCODE = 1
```

Set the dialcode of your country e.g. 44 for UK, 1 for US
3.2.4 4. Prefixes to Ignore

List of prefixes to ignore, these prefixes are removed from the phone number prior to analysis. In cases where customers dial 9 for an outside line, 9, 90 or 900 may need to be removed as well to ensure accurate reporting:

```
PREFIX_TO_IGNORE = "+,0,00,0000,00000,011,55555,99999"
```

3.2.5 Examples

So for the USA, to cope with 10 or 11 digit dialling, PN_MAX_DIGITS would be set to 10, and LOCAL_DIALCODE set to 1. Thus 10 digit numbers would have a 1 added, but 11 digit numbers are left untouched.

In the UK, the number of significant digits is either 9 or 10 after the “0” trunk code. So to ensure that all UK numbers had 44 prefixed to them and the single leading 0 removed, the prefixes to ignore would include 0, the PN_MAX_DIGITS would be set to 10, and the LOCAL_DIALCODE would be 44.

In Spain, where there is no “0” trunk code, and the length of all numbers is 9, then the PN_MAX_DIGITS would be set to 9, and the LOCAL_DIALCODE set to 34.

NB: After changing this file, then both celery and apache should be restarted.

3.3 Configuration for Asterisk

3.3.1 Import configuration for Asterisk

Review your database settings and ensure the second database exists and that is configured correctly:

```
# DATABASE SETTINGS
# ================

DATABASES = {
    'default': {'ENGINE': 'django.db.backends.postgresql_psycopg2', 'NAME': 'cdrstats-billing', 'USER': 'postgres', 'PASSWORD': 'password', 'HOST': 'localhost', 'PORT': '5433', 'OPTIONS': {}},
    'import_cdr': {'ENGINE': 'django.db.backends.postgresql_psycopg2', 'NAME': 'cdr-pusher', 'USER': 'postgres', 'PASSWORD': 'password', 'HOST': 'localhost', 'PORT': '5433', 'OPTIONS': {'autocommit': True}},

    }
```

You will need to push your CDRs from Asterisk CDR datastore to a friendly CDR-Stats ‘import_cdr’ database. To help on this job we created CDR-Pushed, please visit the website and the instructions there to install and configure CDR-Stats correctly: https://github.com/areski/cdr-pusher
3.4 Realtime configuration for Asterisk

The Asterisk Manager settings allow CDR-Stats to retrieve Realtime information to show the number of concurrent calls both in realtime and historically.

In Asterisk, add a new user in manager.conf, or one of its #include’s for CDR-Stats. Further information about Asterisk Manager can be found here: http://www.voip-info.org/wiki/view/Asterisk+config+manager.conf

The collection of realtime information is done via Collectd (https://collectd.org/) and InfluxDB (http://influxdb.com/).

3.5 Configuration for FreeSWITCH

3.5.1 Import configuration for FreeSWITCH

Review your database settings and ensure the second database exists and that is configured correctly:

```python
# DATABASE SETTINGS
# ================

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'cdrstats-billing',
        'USER': 'postgres',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '5433',
        'OPTIONS': {
            # Postgresql Autocommit 'autocommit': True,
        }
    },
    'import_cdr': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'cdr-pusher',
        'USER': 'postgres',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '5433',
        'OPTIONS': {
            'autocommit': True,
        }
    }
}
```

You will need to push your CDRs from Asterisk CDR datastore to a friendly CDR-Stats `import_cdr` database. To help on this job we created CDR-Pushed, please visit the website and the instructions there to install and configure CDR-Stats correctly: https://github.com/areski/cdr-pusher

3.6 Realtime configuration for FreeSWITCH

The FreeSWITCH Event Socket Library allow CDR-Stats to retrieve Realtime information to show the number of concurrent calls both in realtime and historically.

The collection of realtime information is done via Collectd (https://collectd.org/) and InfluxDB (http://influxdb.com/).

CDR-Stats can get CDR from both Freeswitch and Asterisk, or a combination of both. There is other Telco Switches supported please contact us for further information.
3.7 Resetting CDR Data

Sometimes, some experimentation is required to get the optimum settings for country reporting, to achieve this the data is removed from CDR-Stats and re-imported from the CDR data store.

3.7.1 1. Stop Celery

Stop CDR-Stats celery:

```
/etc/init.d/cdr-stats-celeryd stop
```

3.7.2 2. Empty the CDR-Stats dbshell

Enter in the virtualenv and launch dbshell the following commands:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py dbshell
```

Now you are connected on PostgreSQL cli, this is the internal database of CDR-Stats.

The following command will delete all the CDRs, make sure you know what are you doing here and that your CDRs are backed in the upstream CDR data store.

```
$ DELETE FROM voip_cdr;
```

CTRL-D exits the console.

3.7.3 3. Flag the CDR records for reimport

Enter in the virtualenv and launch dbshell the following commands:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py dbshell --database=import_cdr
```

Enter the postgresql password you can find in `settings_local_py` conf file.

Now you are connected on PostgreSQL cli, you can flag CDRs for reimport:

```
$ UPDATE cdr_import SET imported=FALSE;
```

CTRL-D exits the console.

3.7.4 4. Start Celery

Start CDR-Stats celery:

```
/etc/init.d/cdr-stats-celeryd start
```

3.7.5 5. Wait while the CDR are re-imported

Go to the diagnostic page to check if CDR-Stats is correctly configured and if data are being imported.
3.8 Celery Configuration

3.8.1 After installing Broker (Redis or Rabbitmq)

1. Redis Settings

This is a configuration example for Redis.

```python
# Redis Settings
CARROT_BACKEND = "ghettoq.taproot.Redis"

BROKER_HOST = "localhost"  # Maps to redis host.
BROKER_PORT = 6379         # Maps to redis port.
BROKER_VHOST = "0"         # Maps to database number.

CELERY_RESULT_BACKEND = "redis"
REDIS_HOST = "localhost"
REDIS_PORT = 6379
REDIS_DB = 0
#REDIS_CONNECT_RETRY = True
```

2. Rabbitmq Settings

This is a configuration example for Rabbitmq.

```python
BROKER_HOST = "localhost"
BROKER_PORT = 5672
BROKER_USER = "root"
BROKER_PASSWORD = "root"
BROKER_VHOST = "localhost"

CELERY_RESULT_BACKEND = "amqp"
```

3.8.2 Launch celery/celerybeat in debug mode

To run celeryd

```bash
$ python manage.py celeryd -E -l debug
```

To run celerybeat

```bash
$ python manage.py celerybeat --schedule=/var/run/celerybeat-schedule
```

To run both

```bash
$ python manage.py celeryd -E -B -l debug
```

3.8.3 Running celery/celerybeat as a daemon (Debian/Ubuntu)

To configure celeryd as a daemon, it is necessary to configure the location of celeryconfig

```bash
$ cd install/celery-init/etc/default/
```
1. Open celeryd in text editor & change the following variables
   
   Configuration file: /etc/default/celeryd
   
   Init script: celeryd.
   
   Usage: /etc/init.d/celeryd {start|stop|force-reload|restart|try-restart|status}:
   
   ```
   # Where to chdir at start
   CELERYD_CHDIR="/path/to/cdr-stats/"
   
   # Path to celeryd
   CELERYD="/path/to/cdr-stats/manage.py celeryd"
   
   # Extra arguments to celeryd
   CELERYD_OPTS="--time-limit=300"
   
   # Name of the celery config module.
   CELERY_CONFIG_MODULE="celeryconfig"
   
   # Extra Available options
   # %n will be replaced with the nodename.
   # Full path to the PID file. Default is /var/run/celeryd.pid.
   CELERYD_PID_FILE="/var/run/celery/%n.pid"
   
   # Full path to the celeryd log file. Default is /var/log/celeryd.log
   CELERYD_LOG_FILE="/var/log/celery/%n.log"
   
   # User/Group to run celeryd as. Default is current user.
   # Workers should run as an unprivileged user.
   CELERYD_USER="celery"
   CELERYD_GROUP="celery"
   ```
   
2. Open celeryd (for periodic task) in text editor & add the following variables
   
   Configuration file: /etc/default/celerybeat or /etc/default/celeryd
   
   Init script: celerybeat
   
   Usage: /etc/init.d/celerybeat {start|stop|force-reload|restart|try-restart|status}:
   
   ```
   # Path to celerybeat
   CELERYBEAT="/path/to/cdr-stats/manage.py celerybeat"
   
   # Extra arguments to celerybeat
   CELERYBEAT_OPTS="--schedule=/var/run/celerybeat-schedule"
   ```
   
3. Copy the configuration file & init scripts to /etc dir:
   
   ```
   $ cp etc/default/celeryd /etc/default/
   $ cp etc/init.d/celeryd /etc/init.d/
   $ cp etc/init.d/celerybeat /etc/init.d/
   ```
   
4. Run/Start or Stop celery as a daemon:
   
   ```
   $ /etc/init.d/celeryd start or stop
   $ /etc/init.d/celerybeat start or stop
   ```
3.8.4 Troubleshooting

If celeryd will not start as a daemon, try running it in verbose mode:

```bash
$ sh -x /etc/init.d/celeryd start
$ sh -x /etc/init.d/celerybeat start
```

3.9 ACL Control

One of the benefits of CDR-Stats is ACL access, allowing numerous people to access CDR-Stats each viewing their own CDR with permissions assigned to allow viewing different parts of the interface.

3.9.1 Add Customer

To add a new user, enter the admin screen and Add Customer. Enter a username and password, (twice for authentication), optionally add address details, then enter the accountcode of the customer which corresponds to the accountcode that is delivered in the CDR. When done, click save, and the customer details will be saved and the page reloaded and now displays the user permissions available.

Permissions can be added individually by selecting the permission and then pressing the right arrow to move the permission from the left field to the right field. When done, click save. The permissions to assign to the user are those beginning with user_profile and cdr_alert.

3.9.2 Group Permissions

When you have many customers who are all to have the same permissions, you can add a group, assign the group the desired permissions, then add the customer to the group.

From the admin screens, Click add group, give it a name, assign permissions then save. Finally edit the customer, select the groups to which the customer will belong, then click save. The customer will then inherit permissions from their group.
Troubleshooting

4.1 Where to find the log files

All the logs are centralized into one single directory /var/log/cdr-stats/

- **cdr-stats.log**: All the logger events from Django
- **cdr-stats-db.log**: This contains all the Database queries performed by the UI
- **gunicorn_cdr_stats.log**: All the logger events from Gunicorn
- **djcelery_error.log**: This contains celery activity
- **djcelerybeat_error.log**: This contains celerybeat activity

4.2 Run in debug mode

Make sure services are stopped first:

```
$ /etc/init.d/cdrstats-celeryd stop
```

Then run in debug mode:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py celeryd -EB --loglevel=DEBUG
```

4.3 Celerymon

- [https://github.com/ask/celerymon](https://github.com/ask/celerymon)

Running the monitor:
Start celery with the --events option on, so celery sends events for celerymon to capture::

```
$ workon cdr-stats $ cd /usr/share/cdrstats/ $ python manage.py celeryd -E
```

Run the monitor server:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py celerymon
```

However, in production the monitor is best run in the background as a daemon:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py celerymon --detach
```

For a complete listing of the command line arguments available, with a short description, use the help command:

```
$ workon cdr-stats
$ cd /usr/share/cdrstats/
$ python manage.py help celerymon
```

Visit the webserver celerymon stats by going to: http://localhost:8989
CHAPTER 5

User Guide

Contents:

5.1 Overview

CDR-Stats is a web based application built on top of the Django Web framework, which uses PostgreSQL as the CDR data store.

Celery (http://celeryproject.org/) is an asynchronous task queue/job queue based on distributed message. It is used to build our backend system to monitor CDR, detect unusual activity, and react by sending alert email.

CDR Stats Management Features

- Multi-tenant design that allows call detail records from multiple switches or PBX systems.
- Custom alarm triggers can be set to email the administrator for a range of conditions including unusual average call durations, failed calls, and unexpected destinations called.
- Graphical tools help detect unusual call patterns which may indicate suspicious or fraudulent activity.
- Import Call Detail Records in CSV format
- Configure Switches for import
- Create Customer and assign accountcode
- Configure alert to detect unusual increase/decrease of Traffic

CDR Stats Customer Portal Features

- Password management
- Call Details Record
- Monthly, Daily, Hourly Call reporting
- Impact Reporting
- Country Reporting
- Realtime Reporting of calls in progress
- View Fraudulent Calls
- Concurrent Call Statistic
- Configure Mail Reporting
- Top 10 destination Traffic
• Export to CSV
• Automated daily reporting.

5.2 How to use CDR-Stats

CDR-Stats has two main areas, the admin screen and the customer portal. The admin and customer areas are described in detail in the following pages.

CDR-Stats has been designed to be responsive, that is to say the layout changes depending on the size and resolution of the browser viewing the pages.

5.3 Admin Panel

http://localhost:8000/admin/

The Admin section allows you to create administrators who have access to the admin screens. Levels of access can be set.

5.3.1 Screenshot with Features

Dashboard

Dashboard page for the admin interface after successful login with superuser credentials

Alarm

The alarm list will be displayed from the following URL. You can add a new alarm by clicking Add alarm and adding the name of the alarm and its description. Also from the alarm list, click on the alarm that you want to update.

URL:
To Add/Update alarm

URL:
- http://localhost:8000/admin/cdr_alert/alarm/1/

Alarm-report

The alarm-report will be displayed from the following URL.

URL:
- http://localhost:8000/admin/cdr_alert/alarmreport/

To Add/Update alarm-report

URL:
- http://localhost:8000/admin/cdr_alert/alarmreport/1/
**Blacklist**

The blacklist will be displayed from the following URL. You can add a new blacklist by clicking Blacklist by country and selecting the country name and its prefixes. Also, from the blacklist, click on the blacklist that you want to update.

**URL:**

- http://localhost:8000/admin/cdr_alert/blacklist/

**Blacklist by country**

**Country:** Spain

[Select country]
Whitelist

The whitelist will be displayed from the following URL. You can add a new Whitelist by clicking Whitelist by country and selecting the country name and its prefixes, Also from the whitelist, click on the blacklist that you want to update.

URL:

- http://localhost:8000/admin/cdr_alert/whitelist/

![Whitelist by country](image)

Select Whitelist to change

<table>
<thead>
<tr>
<th>ID</th>
<th>Phonenumber prefix</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Whitelist by country

Country: Afghanistan

Select country

- [ ] Select all prefixes

- [ ] 93
- [ ] 937
- [ ] 3341
- [ ] 9379
- [ ] 9375
- [ ] 9377
- [ ] 9378
- [ ] 9379

Blacklist the selected prefixes  Blacklist the selected country

Alert-remove-prefix

The alert remove prefix will be displayed from the following URL. You can add a new remove prefix by clicking Add alert remove prefix and selecting the remove prefix, Also from the alert remove prefix, click on the remove prefix that you want to update.

URL:

- http://localhost:8000/admin/cdr_alert/alertremoveprefix/

![Alert-remove-prefix](image)

Select Alert Remove Prefix to change

<table>
<thead>
<tr>
<th>ID</th>
<th>Label</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>93375</td>
</tr>
</tbody>
</table>

To Add/Update alert-remove prefix

URL:

- http://localhost:8000/admin/cdr_alert/alertremoveprefix/1/
Switch

URL:

- http://localhost:8000/admin/cdr/switch/

HangupCause

URL:

- http://localhost:8000/admin/cdr/hangupcause/

CDR View

URL:

5.4 Customer Panel

User Interface :
This application provides a user interface...

http://localhost:8000/

- *Screenshot with Features*

5.4.1 Screenshot with Features

Index
Index page for the customer interface after successful login with user credentials
Dashboard

The dashboard displays a graphical representation of the last 24 hours calls, call status statistics and calls by country, either aggregated for all switches, or selectable by switch.

URL:

- http://localhost:8000/dashboard/
CDR-View

Call detail records listed in table format which can be exported to CSV file.
Advanced Search allows further filtering and searching on a range of criteria
The Report by Day shows a graphical illustration of the calls, minutes and average call time.
URL:

- http://localhost:8000/cdr_view/

Calls Details Record - 1st April 2012 to 30th April 2012

<table>
<thead>
<tr>
<th>Call-date</th>
<th>Cid</th>
<th>Destination</th>
<th>Dur</th>
<th>Bill</th>
<th>Hangup cause</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 24, 2012, 6:22 a.m.</td>
<td>78191200 - 78191209</td>
<td>1843145</td>
<td>0:00</td>
<td>0:00</td>
<td>NORMAL_CLEARING</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:21 a.m.</td>
<td>57682127 - 57682127</td>
<td>4414367</td>
<td>0:00</td>
<td>0:00</td>
<td>USER_BUSY</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:19 a.m.</td>
<td>36267763 - 36267765</td>
<td>04121991</td>
<td>0:00</td>
<td>0:00</td>
<td>NORMAL_CLEARING</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:19 a.m.</td>
<td>56404247 - 56404247</td>
<td>5523623</td>
<td>01:19</td>
<td>0:21</td>
<td>USER_BUSY</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:17 a.m.</td>
<td>57250890 - 57250890</td>
<td>034430587</td>
<td>0:00</td>
<td>0:00</td>
<td>NO_USER_RESPONSE</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:15 a.m.</td>
<td>13780718 - 13780718</td>
<td>5557424</td>
<td>0:00</td>
<td>0:00</td>
<td>NO_USER_RESPONSE</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:13 a.m.</td>
<td>57236105 - 57236105</td>
<td>45175147</td>
<td>0:00</td>
<td>0:00</td>
<td>NO_ANSWER</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 6:09 a.m.</td>
<td>74008177 - 74008177</td>
<td>3970392</td>
<td>0:21</td>
<td>0:20</td>
<td>USER_BUSY</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 5:55 a.m.</td>
<td>30034618 - 30034618</td>
<td>-939081997</td>
<td>0:00</td>
<td>0:00</td>
<td>CALL_REJECTED</td>
<td>1000</td>
</tr>
<tr>
<td>April 24, 2012, 5:59 a.m.</td>
<td>59425163 - 59425163</td>
<td>184086626</td>
<td>0:30</td>
<td>0:21</td>
<td>NO_ANSWER</td>
<td>1000</td>
</tr>
</tbody>
</table>
CDR-Overview

A pictorial view of calls with call-count or call-duration from any date or date-range

**URL:**
- http://localhost:8000/cdr_overview/
CDR-Hourly-Report

An hourly pictorial view of calls with call-count & call-duration. You can compare different dates

URL:
  • http://localhost:8000/hourly_report/
CDR-Country-Report

A pictorial view of all calls by country with the 10 most called countries in a pie chart.

URL:
Mail-Report

A list of the last 10 calls of the previous day, along with total calls, a breakdown of the call status, and the top 5 countries called.

This report is emailed automatically, email recipients can be set up in the admin section or by adding an email address in the “Email to send a report” field in the Mail Report section.

**URL:**
Concurrent-call-report

A report of concurrent calls. The statistics are collated from the realtime report, not from the CDR.

URL:

Realtime-Report

Realtime monitoring of the traffic on the connected telecoms servers, Freeswitch and Asterisk are supported.

URL:

- http://localhost:8000/cdr_realtime/

World Map Report

A distribution map of all calls / durations by country. You can select date criteria and on mouse over on the world map you can get information about each country.
URL:

- http://localhost:8000/world_map/

**World Map Report - 1st Nov. 2012 to 30th Nov. 2012**

<table>
<thead>
<tr>
<th>Country</th>
<th>Calls</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgium</td>
<td>4649</td>
<td>3906.20 minutes</td>
</tr>
<tr>
<td>Italy</td>
<td>4473</td>
<td>3820.08 minutes</td>
</tr>
<tr>
<td>Cuba</td>
<td>4385</td>
<td>3732.06 minutes</td>
</tr>
<tr>
<td>Denmark</td>
<td>4390</td>
<td>3724.34 minutes</td>
</tr>
<tr>
<td>Spain</td>
<td>6410</td>
<td>3718.36 minutes</td>
</tr>
<tr>
<td>Brazil</td>
<td>6657</td>
<td>3703.38 minutes</td>
</tr>
<tr>
<td>India</td>
<td>4491</td>
<td>3659.23 minutes</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>4475</td>
<td>3502.09 minutes</td>
</tr>
<tr>
<td>United States</td>
<td>3619</td>
<td>2453.23 minutes</td>
</tr>
<tr>
<td>Canada</td>
<td>3618</td>
<td>2429.56 minutes</td>
</tr>
</tbody>
</table>

Powered by CDR-Stats - Call Monitoring & Analytics Software

**Alert Settings**

URL:

- http://localhost:8000/alert/

5.4. Customer Panel
**Alert Report**

**URL:**

**Destination Control**

**URL:**
Diagnostic CDR-Stats

URL:

- http://localhost:8000/diagnostic/

- CDR Backend [127.0.0.1] connected successfully.

Rates

voip call rates.

URL:
Call Simulator

voip call simulator.

URL:

- http://localhost:8000/simulator/

Daily report of Billed call

Daily report of Billed call.

URL:

5.4. Customer Panel
PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance.

PostgreSQL provides few interesting features that make a perfect pick for CDR-Stats:

- **Materialized view** (http://www.postgresql.org/docs/9.4/static/rules-materializedviews.html), those views contain the results of queries, it’s very ideal for aggregation view, they also can be refreshed since PG 9.4 without locking.

- **Json Types** (http://www.postgresql.org/docs/9.4/static/datatype-json.html), are for storing JSON (JavaScript Object Notation) data, this field is ideal to store none structured data. CDR-Stats aggregate data from several type of telco switches in which the type of data received can vary drastically.

### 6.1 Materialized views

We created 2 Materialized views to help on our reporting job, here is the schema structure of those 2 views:

- Materialized View CREATE MATERIALIZED VIEW matv_voip_cdr_aggr_hour AS

  ```sql
  SELECT date_trunc('hour', starting_date) as starting_date, country_id, switch_id, cdr_source_type, hangup_cause_id, user_id, count(*) AS nbcalls, sum(duration) AS duration, sum(billsec) AS billsec, sum(buy_cost) AS buy_cost, sum(sell_cost) AS sell_cost
  FROM voip_cdr
  GROUP BY date_trunc('hour', starting_date), country_id, switch_id, cdr_source_type, hangup_cause_id, user_id;
  ```

- Create index on Materialized view CREATE UNIQUE INDEX matv_voip_cdr_aggr_hour_date ON matv_voip_cdr_aggr_hour (starting_date, country_id, switch_id, cdr_source_type, hangup_cause_id);

- Materialized View CREATE MATERIALIZED VIEW matv_voip_cdr_aggr_min AS

  ```sql
  ```
SELECT date_trunc('minute', starting_date) as starting_date, country_id, switch_id, cdr_source_type, hangup_cause_id, user_id, count(*) AS nbcalls, sum(duration) AS duration, sum(billsec) AS billsec, sum(buy_cost) AS buy_cost, sum(sell_cost) AS sell_cost
FROM voip_cdr
GROUP BY date_trunc('minute', starting_date), country_id, switch_id, cdr_source_type, hangup_cause_id, user_id;

-- Create index on Materialized view CREATE UNIQUE INDEX matv_voip_cdr_aggr_min_date
ON matv_voip_cdr_aggr_min (starting_date, country_id, switch_id, cdr_source_type, hangup_cause_id);

You can drop those views with:

-- Drop Materialized View
DROP MATERIALIZED VIEW matv_voip_cdr_aggr_hour;

-- Drop Materialized View
DROP MATERIALIZED VIEW matv_voip_cdr_aggr_min;

You can refresh the view as follow, using concurrently we ensure to not lock the view:

# Refresh without lock
REFRESH MATERIALIZED VIEW CONCURRENTLY matv_voip_cdr_aggr_hour;

# Refresh without lock
REFRESH MATERIALIZED VIEW CONCURRENTLY matv_voip_cdr_aggr_min;

The update of the Materialized view is done periodically by a celery task using the above commands “REFRESH MATERIALIZED VIEW”.

7.1 Prerequisites

To fully understand this project, developers will need to have a advanced knowledge of:

- Django: http://www.djangoproject.com/
- Celery: http://www.celeryproject.org/
- Python: http://www.python.org/
- Freeswitch: http://www.freeswitch.org/
- Freeswitch: http://www.asterisk.org/

7.2 Coding Style & Structure

7.2.1 Style

Coding follows the PEP 8 Style Guide for Python Code.

7.2.2 Structure

The CDR-Stats directory:

|-- api
|-- |-- api_playground
|-- cdr
| `-- management
| `-- template
| `-- fixtures
|-- cdr_alert
| `-- management
| `-- fixtures
|-- frontend
| `-- management
| `-- fixtures
|-- user_profile
| `-- management
| `-- fixtures
|-- static
```
|-- cdr
| |-- css
| |-- js
| |-- icons
|  `-- images
|-- resources - This area is used to hold media files
 `-- templates - This area is used to override templates
    |-- admin
    |-- admin_tools
    |-- api_browser
    `-- frontend
```
7.3 Objects Description

7.3.1 Switch

7.3.2 HangupCause

7.3.3 UserProfile

7.3.4 Alarm

7.3.5 AlertRemovePrefix

7.3.6 AlarmReport

7.3.7 Blacklist

7.3.8 Whitelist

7.3.9 VoIPPlan

7.3.10 BanPlan

7.3.11 VoIPPlan_BanPlan

7.3.12 BanPrefix

7.3.13 VoIPRetailPlan

7.3.14 VoIPPlan_VoIPRetailPlan

7.3.15 VoIPRetailRate

7.3.16 VoIPCarrierPlan

7.3.17 VoIPCarrierRate

7.3.18 VoIPPlan_VoIPCarrierPlan

7.4 Database Design

The current database schema is shown below:
7.5 CDR-Stats Views

7.5.1 cdr_view

7.5.2 cdr_detail

7.5.3 cdr_dashboard

7.5.4 cdr_overview

7.5.5 cdr_realtime

7.5.6 cdr_daily_comparison

7.5.7 cdr_concurrent_calls

7.5.8 world_map_view
7.5.9 mail_report

7.5.10 customer_detail_change

7.5.11 alarm_list

7.5.12 alarm_add

7.5.13 alarm_del

7.5.14 alarm_change

7.5.15 alarm_test

7.5.16 alert_report

7.5.17 trust_control

7.5.18 index

7.5.19 diagnostic
7.5.20 login_view

7.5.21 logout_view

7.5.22 pleaselog

7.5.23 voip_rates

7.5.24 export_rate

7.5.25 simulator

7.5.26 billing_report

7.5.27 cust_password_reset

7.5.28 cust_password_reset_done

7.5.29 cust_password_reset_confirm

7.5.30 cust_password_reset_complete
7.6 CDR-Stats Tasks

7.6.1 sync_cdr_pending

7.6.2 chk_alarm

7.6.3 blacklist_whitelist_notification

7.6.4 send_cdr_report

7.6.5 RebillingTask

7.6.6 ReaggregateTask

7.7 Test Case Descriptions

7.7.1 Requirement

**Run/Start Celery:**

```bash
$ /etc/init.d/celery start
```

or:

```bash
$ python manage.py celeryd -l info
```

**Run/Start Redis:**

```bash
$ /etc/init.d/redis-server start
```

7.7.2 How to Run Tests

1. **Run Full Test Suit:**

   ```bash
   $ python manage.py test --verbosity=2
   ```

3. **Run CDRStatsAdminInterfaceTestCase:**

   ```bash
   $ python manage.py test cdr.CDRStatsAdminInterfaceTestCase --verbosity=2
   ```

4. **Run CDRStatsCustomerInterfaceTestCase:**

   ```bash
   $ python manage.py test cdr.CDRStatsCustomerInterfaceTestCase --verbosity=2
   ```

7.8 Javascript Files

- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.
• **NVD3** is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you. This is a very young collection of components, with the goal of keeping these components very customizeable, staying away from your standard cookie cutter solutions.

• **Bootstrap** is sleek, intuitive, and powerful front-end framework for faster and easier web development.

• **Bootbox** is a small JavaScript library which allows you to create programmatic dialog boxes using Twitter’s Bootstrap modals, without having to worry about creating, managing or removing any of the required DOM elements or JS event handlers.

• **Bootstrap-datepicker** Datepicker for Bootstrap
CHAPTER 8

API Reference

Contents:

8.1 SwitchSerializer

8.2 VoIPRateList

8.3 VoipCallResource

Testing console of APIs:

**URL**: http://127.0.0.1:8000/api-explorer/

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hangupcause</td>
</tr>
<tr>
<td>2</td>
<td>Switch</td>
</tr>
<tr>
<td>3</td>
<td>Cdr</td>
</tr>
</tbody>
</table>

**Powered by CDR-Stats - Call Monitoring & Analytics Software**

To test individual api, click on one api from the api list and you will get a similar screen as follows:

**URL**: http://127.0.0.1:8000/api-explorer/switch/
Switch API Playground

/switch/

This resource allows you to manage switches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/v1/switch/</td>
<td>Returns all switches</td>
</tr>
</tbody>
</table>

**GET** /api/v1/switch/

Request

GET /api/v1/switch/
Content-Type: application/json; charset=utf-8

Response Status
OK (200)

Response Headers
Date: Fri, 19 Oct 2012 10:23:46 GMT
Server: WSGIServer/0.1 Python/2.7.3
Vary: Accept-Language, Cookie
Content-Type: application/json; charset=utf-8
Content-Language: en
Cache-Control: no-cache

Response Body
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 4}, "objects": [{"id": "1", "ipaddress": "127.0.0.1", "key_uuid": "c80445f0-183f-11e2-964f-060e2925d15f", "name": "127.0.0.1", "resource_uri": "/api/v1/switch/1/"}]

Give feedback about this response

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/v1/switch/{switch-id}/</td>
<td>Returns a specific switch</td>
</tr>
</tbody>
</table>

**GET** /api/v1/switch/{switch-id}/

**POST** /api/v1/switch/

Creates new switch

Data Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>localhost</td>
</tr>
<tr>
<td>ipaddress</td>
<td>192.168.1.4</td>
</tr>
</tbody>
</table>

**POST** /api/v1/switch/{switch-id}/

Update switch

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td>/api/v1/switch/{switch-id}/</td>
<td>Update switch</td>
</tr>
</tbody>
</table>

**DELETE** /api/v1/switch/{switch-id}/

Delete switch

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/api/v1/switch/{switch-id}/</td>
<td>Delete switch</td>
</tr>
</tbody>
</table>
9.1 Community Code of Conduct

Members of our community need to work together effectively, and this code of conduct lays down the ground rules for our cooperation.

Please read the following documentation about how the CDR-Stats Project functions, coding styles expected for contributions, and the community standards we expect everyone to abide by.


9.1.1 Be considerate.

Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and we expect you to take those consequences into account when making decisions. Even if it’s not obvious at the time, our contributions to CDR-Stats will impact the work of others. For example, changes to code, infrastructure, policy, documentation and translations during a release may negatively impact others work.

9.1.2 Be respectful.

The CDR-Stats community and its members treat one another with respect. Everyone can make a valuable contribution to CDR-Stats. We may not always agree, but disagreement is no excuse for poor behaviour and bad manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It’s important to remember that a community where people feel uncomfortable or threatened is not a productive one. We expect members of the CDR-Stats community to be respectful when dealing with other contributors as well as with people outside the CDR-Stats project and with users of CDR-Stats.
9.1.3 Be collaborative.

Collaboration is central to CDR-Stats and to the larger free software community. We should always be open to collaboration. Your work should be done transparently and patches from CDR-Stats should be given back to the community when they are made, not just when the distribution is released. If you wish to work on new code for existing upstream projects, at least keep those projects informed of your ideas and progress. It may not be possible to get consensus from upstream, or even from your colleagues about the correct implementation for an idea, so don’t feel obliged to have that agreement before you begin, but at least keep the outside world informed of your work, and publish your work in a way that allows outsiders to test, discuss and contribute to your efforts.

9.1.4 When you disagree, consult others.

Disagreements, both political and technical, happen all the time and the CDR-Stats community is no exception. It is important that we resolve disagreements and differing views constructively and with the help of the community and community process. If you really want to go a different way, then we encourage you to make a derivative distribution or alternate set of packages that still build on the work we’ve done to utilise as common a core as possible.

9.1.5 When you are unsure, ask for help.

Nobody knows everything, and nobody is expected to be perfect. Asking questions avoids many problems down the road, and so questions are encouraged. Those who are asked questions should be responsive and helpful. However, when asking a question, care must be taken to do so in an appropriate forum.

9.1.6 Step down considerately.

Developers on every project come and go and CDR-Stats is no different. When you leave or disengage from the project, in whole or in part, we ask that you do so in a way that minimises disruption to the project. This means you should tell people you are leaving and take the proper steps to ensure that others can pick up where you leave off.

9.2 Reporting a Bug

Bugs can always be described to the Mailing list, but the best way to report an issue and to ensure a timely response is to use the issue tracker.

1. Create a GitHub account.

You need to create a GitHub account to be able to create new issues and participate in the discussion.

2. Determine if your bug is really a bug.

You should not file a bug if you are requesting support. For that you can use the Mailing list.

3. Make sure your bug hasn’t already been reported.

Search through the appropriate Issue tracker. If a bug like yours was found, check if you have new information that could be reported to help the developers fix the bug.

4. Collect information about the bug.

To have the best chance of having a bug fixed, we need to be able to easily reproduce the conditions that caused it. Most of the time this information will be from a Python traceback message, though some bugs might be in design, spelling or other errors on the website/docs/code.

If the error is from a Python traceback, include it in the bug report.
We also need to know what platform you’re running (Windows, OSX, Linux, etc), the version of your Python interpreter, the version of CDR-Stats and related packages that you were running when the bug occurred.

5. Submit the bug.

By default GitHub will email you to let you know when new comments have been made on your bug. In the event you’ve turned this feature off, you should check back on occasions to ensure you don’t miss any questions a developer trying to fix the bug might ask.

### 9.2.1 Issue Trackers

Bugs for a package in the CDR-Stats ecosystem should be reported to the relevant issue tracker.

- Celery: [https://github.com/ask/celery/issues/](https://github.com/ask/celery/issues/)
- Freeswitch: [http://jira.freeswitch.org/secure/Dashboard.jspa](http://jira.freeswitch.org/secure/Dashboard.jspa)

If you are unsure of the origin of the bug you can ask the Mailing list, or just use the CDR-Stats issue tracker.

### 9.3 Coding Style

You should probably be able to pick up the coding style from surrounding code, but it is a good idea to be aware of the following conventions.

- All Python code must follow the PEP-8 guidelines.
- Docstrings must follow the PEP-257 conventions, and use the following style.
  
  ```python
  def method(self, arg):
      """Short description.
      More details.
      """
  ```

  or:

  ```python
  def method(self, arg):
      """Short description."
  ```

  but not this:

  ```python
  def method(self, arg):
      ""
      Short description.
      ""
  ```

- Lines should not exceed 78 columns.
- Wildcard imports must not be used (`from xxx import *`).
10.1 Getting Help

10.1.1 Mailing list

For discussions about the usage, development, and future of CDR-Stats, please join the CDR-Stats mailing list.

10.2 Bug tracker

If you have any suggestions, bug reports or annoyances please report them to our issue tracker at https://github.com/areski/cdr-stats/issues/

10.3 Documentation

The latest documentation with user guides, tutorials and API references is hosted on CDR-Stats website: http://www.cdr-stats.org/documentation/


10.4 Support

Star2Billing S.L. offers consultancy including installation, training and customisation

Website: http://www.star2billing.com

Email: cdr-stats@star2billing.com
10.5 License

This software is licensed under the *MPL 2.0 License*. See the LICENSE file in the top distribution directory for the full license text.
Frequently Asked Questions

11.1 General

11.1.1 What is CDR-Stats?

**Answer:** CDR-Stats is a free and open source web based Call Detail Record analysis application with the ability to display reports and graphs.

11.1.2 Why should I use CDR-Stats?

**Answer:** If you have call detail records from an office PBX, telecoms switch(s), or carrier CDR to analyse then CDR-Stats is a useful tool to analyse the data and look for patterns in the traffic that may indicate problems or potential fraud. Furthermore, CDR-Stats can be configured to send email alerts on detection of unusual activity, as well as send daily reports on traffic.

11.1.3 How to start over, delete CDRs and relaunch the import?

**Answer:** First, stop celery and remove the aggregate data, connect on postgresql and enter the following:

```
$ DROP MATERIALIZED VIEW matv_voip_cdr_aggr_hour;
$ DROP MATERIALIZED VIEW matv_voip_cdr_aggr_min;
```

Then recreate the Materialized View as follow:

```
$ CREATE MATERIALIZED VIEW matv_voip_cdr_aggr_hour AS
SELECT
    date_trunc('hour', starting_date) as starting_date, 
    country_id, 
    switch_id, 
    cdr_source_type, 
    hangup_cause_id, 
    user_id, 
    count(*) AS nbcalls, 
    sum(duration) AS duration,
```
Then, update all your CDRs from ‘import_cdr’ PostgreSQL database to be reimported as we flag them after import:

```sql
$ UPDATE cdr_import SET imported=FALSE;
```

Start Celery, and check CDR are being imported correctly.

### 11.1.4 How to debug mail connectivity?

**Answer:** Use `mail_debug` to test the mail connectivity:

```bash
$ cd /usr/share/cdrstats
$ workon cdr-stats
$ python manage.py mail_debug
```

### 11.1.5 What should I do if I have problems?

**Answer:**

- Review the installation script, and check that services are running.
- Read the documentation contained in the CDR-Stats website.
- Ask a question on the forum.
- Ask a question on the mailing list
- Purchase support from Star2Billing.
CHAPTER 12

Indices and tables

- genindex
- modindex
- search