CDR-Stats Documentation

Release 2.0.0

Arezqui Belaid

January 24, 2013

CONTENTS

Version 2.0

Release 2.0.0

Date January 24, 2013

Contents:

CHAPTER

ONE

GETTING STARTED

Web http://www.cdr-stats.org/

Download http://www.cdr-stats.org/download/

Source https://github.com/Star2Billing/cdr-stats/

Keywords voip, freeswitch, asterisk, django, python, call, reporting, CDR, mongoDB

–

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, SocketIO, Gevent and MongoDB.

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

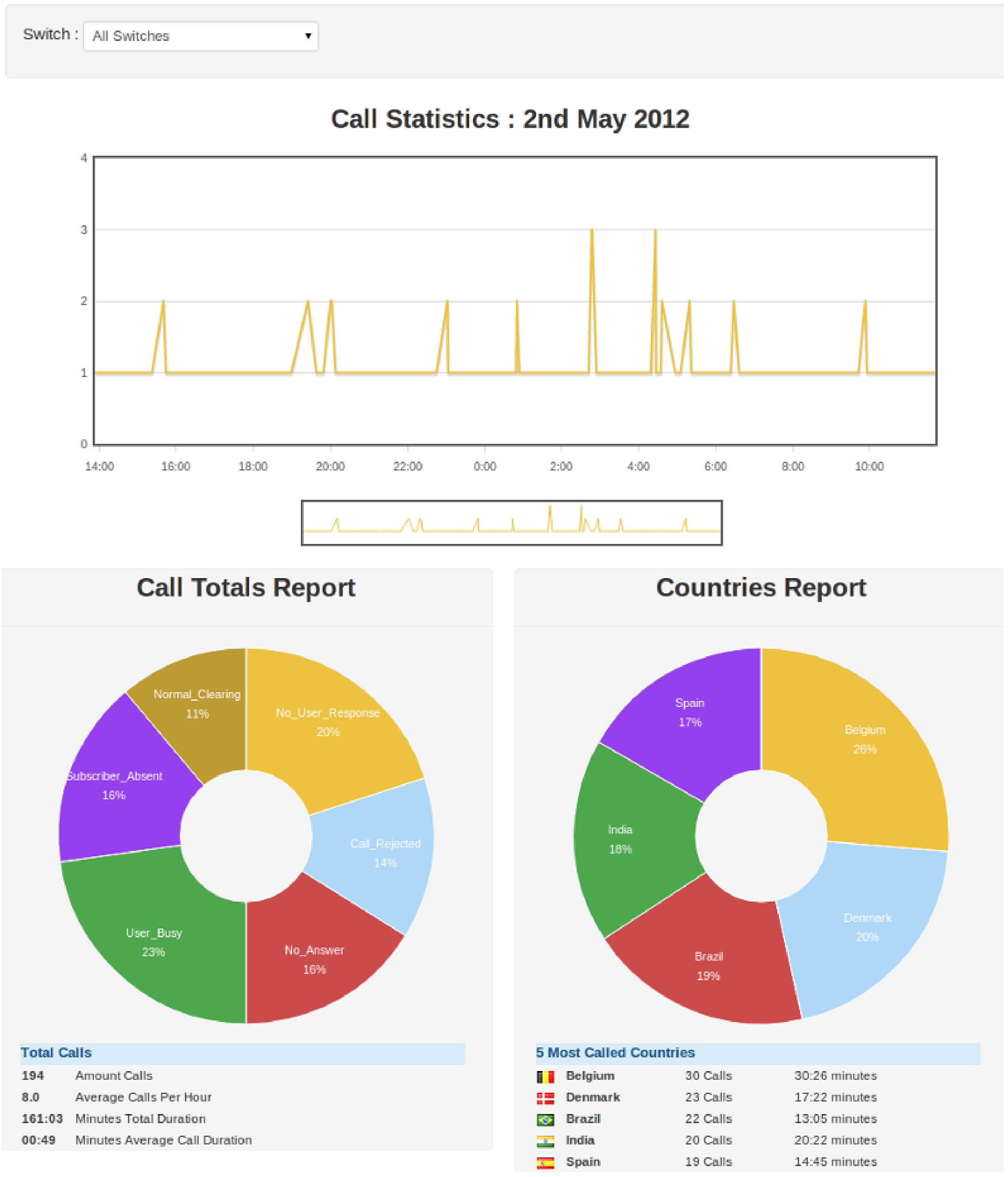
- Mail daily aggregated reports
- Threat Control: Detect abnormal call patterns
- Destination Alerts: Unexpected destination alerts

CDR Stats uses MongoDB, a scalable, high performance database system used to analyse large quantities of CDR data. MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind.

CDR-Stats supports Freeswitch and Asterisk using connectors that get the CDR. Connectors for other switch systems can be built. Additionally CDR-Stats features a CSV upload facility so that CDR from virtually any source can be imported and analysed by CDR-Stats.

1.2 Dashboard

User Dashboard provides realtime monitoring of the most relevant metrics of connected switches.



Total Calls

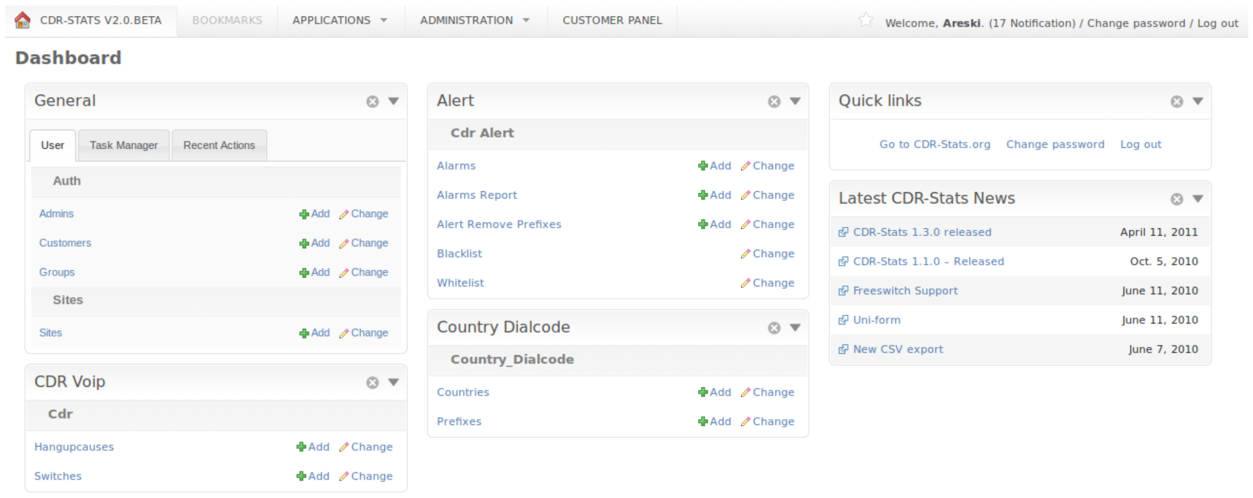
194	Amount Calls
8.0	Average Calls Per Hour
161:03	Minutes Total Duration
00:49	Minutes Average Call Duration

5 Most Called Countries

Belgium	30 Calls	30:26 minutes
Denmark	23 Calls	17:22 minutes
Brazil	22 Calls	13:05 minutes
India	20 Calls	20:22 minutes
Spain	19 Calls	14:45 minutes

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.



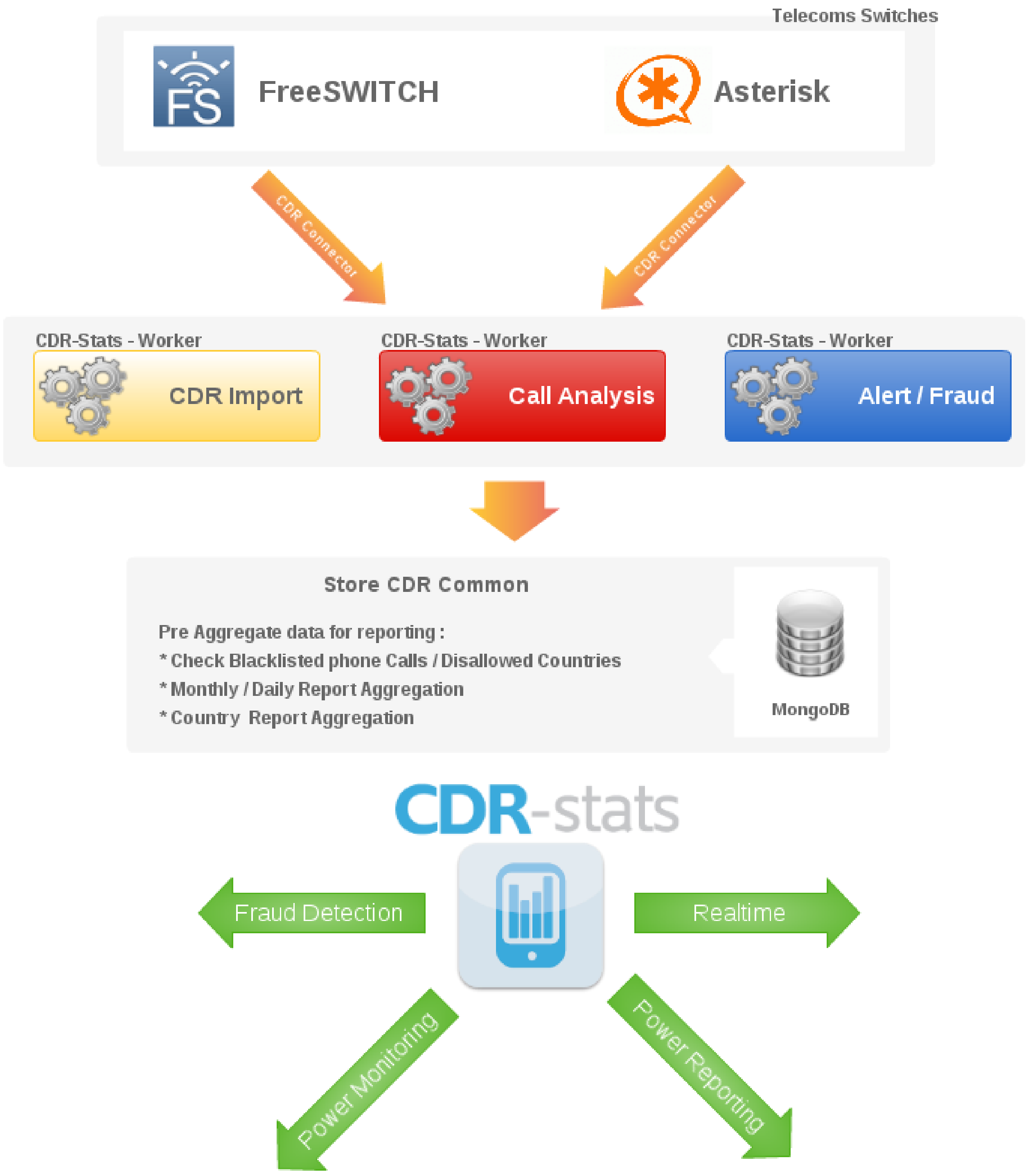
1.4 Architecture

CDR-Stats uses MongoDB as the underlying CDR store. MongoDB 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.



1.5 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.

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

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 in near realtime what calls are going through, can detect errors and failures, and alert the systems administrator is unexpected traffic is noted.

CHAPTER

TWO

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)
- simplejson >= 2.1.3 (Simple, fast, complete, correct and extensible JSON)
- uuid >= 1.30 (UUID object and generation functions)
- wsgiref >= 0.1.2 (Validation support for WSGI)
- django-tastypie (Creating delicious APIs for Django)
- 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
- django-socketio - A Django app providing the features required to use websockets with Django via Socket.IO

The requirements are installed into a virtual environement 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

PIP is a tool for installing and managing Python packages, more information about PIP : http://www.pip-installer.org/en/latest/index.html.

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 webserver 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:

```
$ wget -no-check-certificate https://raw.github.com/Star2Billing/cdr-stats/master/install/install-cdr
$
$ 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 MongoDB in realtime. The web interface for CDR-Stats will be installed on port 8008 and the Websocket on port 9000, so it is neceesary 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:

```
$ wget -no-check-certificate https://raw.github.com/Star2Billing/cdr-stats/master/install/install-cdr
$ 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_mongodb needs to be compiled into Freeswitch. This procedure is described at http://wiki.freeswitch.org/wiki/Mod_cdr_mongodb

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

In freeswitch/conf/autoload_configs/cdr_mongodb.conf.xml

Change:

```
<param name="log-b-leg" value="false"/>
```

To:

```
<param name="log-b-leg" value="true"/>
```

Change:

```
<param name="namespace" value="test.cdr"/>
```

To:

```
<param name="namespace" value="freeswitch_cdr.cdr"/>
```

Then reload the Freeswitch configuration.

Now run the following commands at the console:

```
$ wget -no-check-certificate https://raw.github.com/Star2Billing/cdr-stats/master/install/install-cdr
$
$ 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 Ubuntu 12.04 LTS or CentOS 6.X installation:

```
$ wget -no-check-certificate https://raw.github.com/Star2Billing/cdr-stats/master/install/install-all
$
$ 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

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

or you can use apt-get

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

Running Server

```
$ 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

THREE

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': {
            #Needed on Mysql
            # 'init_command': 'SET storage_engine=INNODB',
            #Postgresql Autocommit
            'autocommit': True,
        }
    }
}
```

Sets up the options to connect to MongoDB Server, this server and database will be used to store the analytic data. There is usually no need to change these settings, unless if your MongoDB server is on a remote server, or if different names are required for the collections:

```
#MONGODB
#=======
MONGO_CDRSTATS = {
    'DB_NAME': 'cdr-stats',
    'HOST': 'localhost',
    'PORT': 27017,
    'CDR_COMMON': 'cdr_common',
    'DAILY_ANALYTIC': 'daily_analytic',
    'MONTHLY_ANALYTIC': 'monthly_analytic',
    'CONC_CALL': 'concurrent_call',
    'CONC_CALL_AGG': 'concurrent_call_aggregate'
}
```

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/cdr-stats/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/cdr-stats/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 = "+,0,00,000,0000,00000,011,55555,99999"
```

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

The asterisk settings may be as follows:

```
#list of CDR Backends to import
CDR_BACKEND = {
    '127.0.0.1': {
        'db_engine': 'mysql',
        'cdr_type': 'asterisk',
        'db_name': 'asteriskcdrdb',
        'table_name': 'cdr',
        'host': 'localhost',
        'port': '',
        'user': 'root',
        'password': 'password',
    },
    #'192.168.1.200': {
        #'db_engine': 'mysql',
        #'cdr_type': 'asterisk',
        #'db_name': 'asteriskcdrdb',
        #'table_name': 'cdr',
        #'host': 'localhost',
        #'port': '',
        #'user': 'root',
        #'password': 'password',
    #},
}
```

To add a new remote Asterisk MySQL CDR store, ensure that there is a connection to the remote MySQL database, then uncomment the new server settings by removing the # and configuring the credentials to connect to the remote Asterisk CDR store.

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.

The settings to configure are:

```
#Asterisk Manager / Used for Realtime and Concurrent calls
ASTERISK_MANAGER_HOST = 'localhost'
ASTERISK_MANAGER_USER = 'cdrstats_user'
ASTERISK_MANAGER_SECRET = 'cdrstats_secret'
```

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

3.5 Configuration for FreeSWITCH

Freeswitch settings are under the CDR_BACKEND section, and should look as follows:

```
CDR_BACKEND = {
    '127.0.0.1': {
        'db_engine': 'mongodb',  # mysql, pgsql, mongodb
        'cdr_type': 'freeswitch',  # asterisk or freeswitch
        'db_name': 'freeswitch_cdr',
        'table_name': 'cdr',  # collection if mongodb
        'host': 'localhost',
        'port': 3306,  # 3306 mysql, 5432 pgsql, 27017 mongodb
        'user': '',
        'password': '',
    },
    #'192.168.1.15': {
    #    'db_engine': 'mongodb',  # mysql, pgsql, mongodb
    #    'cdr_type': 'freeswitch',  # asterisk or freeswitch
    #    'db_name': 'freeswitch_cdr',
    #    'table_name': 'cdr',  # collection if mongodb
    #    'host': 'localhost',
    #    'port': 3306,  # 3306 mysql, 5432 pgsql, 27017 mongodb
    #    'user': '',
    #    'password': '',
    #},
}
```

To connect a new Freeswitch system to CDR-Stats, ensure that port 27017 TCP is ONLY open to the CDR-Stats server on the remote system, then uncomment the settings by removing the #, and configure the IP address and db_name to match those in the mod_cdr_mongodb configuration as described at : http://www.cdr-stats.org/documentation/beginners-guide/howto-installing-on-freeswitch/

CDR-Stats can get CDR from both Freeswitch and Asterisk, or a combination of both.

3.6 Resetting CDR Data

Sometimes, some experimentation is required to get the optimum settings for country reporting, to achieve this the data is removed from MongoDB on CDR-Stats and re-imported from either the Asterisk MySQL CDR store, or the Freeswitch MongoDB CDR-Store.

3.6.1 1. Stop Celery

Stop CDR-Stats celery:

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

3.6.2 2. Empty the CDR-Stats MonoDB data store

Type mongo to enter the MongoDB database then apply the following commands:

```
mongo
use cdr-stats;
db.monthly_analytic.remove({});
db.daily_analytic.remove({});
db.aggregate_world_report.remove({});
db.aggregate_result_cdr_view.remove({});
db.aggregate_hourly_country_report.remove({});
db.cdr_common.remove({});
```

CTRL-D exits the console.

3.6.3 3. Flag the CDR records for reimport

1. With Asterisk and Mysql.

   Go to the CDR database in Asterisk and change the field 'import_cdr' to 0:

   Enter the MySQL console with the following command, changing the credentials and database name to suit your installation:

   ```
   mysql -uasteriskuser -pamp109 asteriskcdrdb
   update cdr SET import_cdr = 0;
   ```

   CTRL-C exits the MySQL

2. With FreeSWITCH and MongoDB.

   Go to the CDR FreeSWITCH MongoDB database, update all the records by setting the 'import_cdr' field to 0:

   ```
   mongo
   use freeswitch_cdr;
   db.cdr.update({"import_cdr" : 1}, { $set : {"import_cdr" : 0}}, { multi: true });
   ```

CTRL-D exits

3.6.4 4. Start Celery

Start CDR-Stats celery:

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

3.6.5 5. Wait while the CDR are re-imported

Go to the diagnostic page to check if the CDR-Backend are correctly configured and if data are being imported.

3.7 Celery Configuration

3.7.1 After installing Broker (Redis or Rabbitmq)

1. Redis Settings

This is a configuration example for Redis.

```
# 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.

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

CELERY_RESULT_BACKEND = "amqp"
```

3.7.2 Launch celery/celerybeat in debug mode

To run celeryd

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

To run celerybeat

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

To run both

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

3.7.3 Running celeryd/celerybeat as a daemon (Debian/Ubuntu)

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

```
$ 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/newfies/"

   # Path to celeryd
   CELERYD="/path/to/newfies/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/newfies/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
   ```

3.7.4 Troubleshooting

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

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

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

3.8 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.

3.8.1 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.

3.8.2 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.

CHAPTER

FOUR

TROUBLESHOOTING

4.1 Where to find the log files

All the logs are centralized into one single directory **/var/log/cdrstats/**

cdrstats-django-db.log : This contains all the Database queries performed by the UI

cdrstats-django.log : All the logger events from Django

err-apache-cdrstats.log : Any apache errors pertaining to CDR-Stats

celery-cdrstats-node1.log : This contains celery activity

4.2 Check if MongoDB is running

Make sure MongoDB is well installed and running:

```
ps auxw | grep mongo
```

You should see something like:

```
mongodb   1184  0.2  0.2 572936  8640 ?        Ssl  Nov25  20:25 /usr/bin/mongod --config /etc/mongoc
```

If the above failed, you might be willing to try to install MongoDB 2.2 manually : http://www.mongodb.org/

If MongoDB is running fine, you can then check if some data has been pulled correctly. Type the following on shell:

```
mongo cdr-stats
```

Tthen on MongoDB CLI:

```
db.cdr_common.findOne();
```

You should see some data, if it's not the case, backend process of CDR-Stats in charge of retrieving your CDRs and pushing them to MongoDB might have some issue. We will recommend to start by checking Celery logs, then if all the configuration to access database are correct, cf file /usr/share/cdr-stats/settings_local.py

4.3 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
```

4.4 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/cdr-stats/
$ 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

FIVE

USER GUIDE

Contents:

5.1 Overview

CDR-Stats is a web based application built on the Django framework, which uses MongoDB as the CDR data stare, and uses MySQL, SQLite or Postgresql for Django framework management and user control.

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 unsual 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 the layout changes depending on the size and resolution of the browser viewing the pages.

5.3 Admin Panel

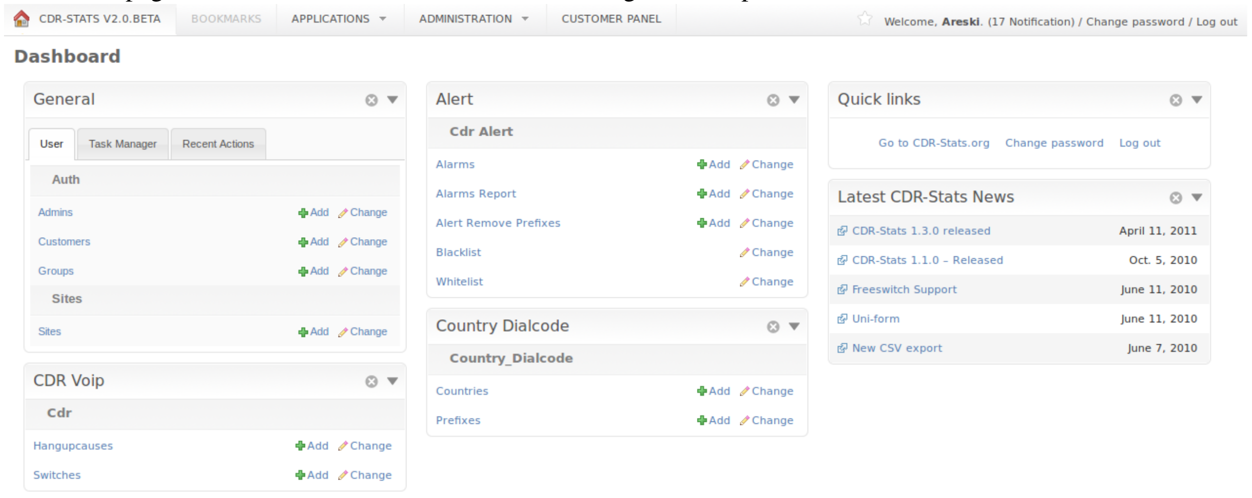
http://localhost:8000/admin/

The Admin section allows you to create administrators who have access 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



Dashboard

General

User | Task Manager | Recent Actions

Auth
- Admins — Add, Change
- Customers — Add, Change
- Groups — Add, Change

Sites
- Sites — Add, Change

CDR Voip

Cdr
- Hangupcauses — Add, Change
- Switches — Add, Change

Alert

Cdr Alert
- Alarms — Add, Change
- Alarms Report — Add, Change
- Alert Remove Prefixes — Add, Change
- Blacklist — Change
- Whitelist — Change

Country Dialcode

Country_Dialcode
- Countries — Add, Change
- Prefixes — Add, Change

Quick links

Go to CDR-Stats.org   Change password   Log out

Latest CDR-Stats News

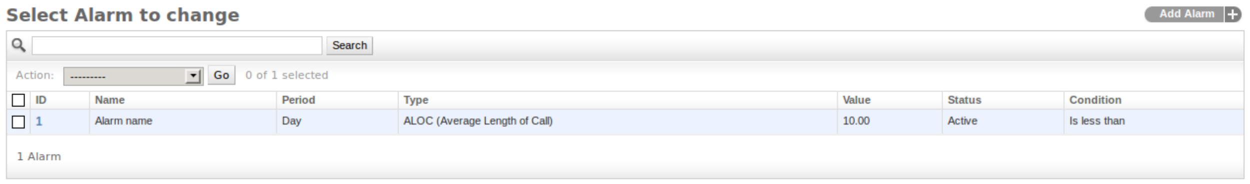
CDR-Stats 1.3.0 released	April 11, 2011
CDR-Stats 1.1.0 - Released	Oct. 5, 2010
Freeswitch Support	June 11, 2010
Uni-form	June 11, 2010
New CSV export	June 7, 2010

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:

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



Select Alarm to change

Add Alarm

Search

Action: --------- Go 0 of 1 selected

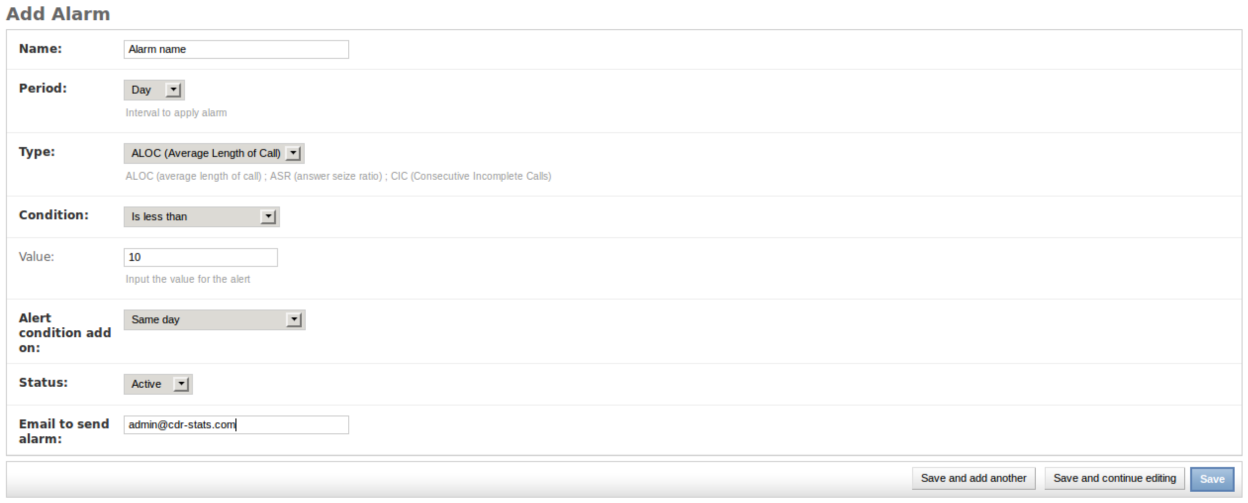
ID	Name	Period	Type	Value	Status	Condition
1	Alarm name	Day	ALOC (Average Length of Call)	10.00	Active	Is less than

1 Alarm

To Add/Update alarm

URL:

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



Add Alarm

Name:	Alarm name
Period:	Day Interval to apply alarm
Type:	ALOC (Average Length of Call) ALOC (average length of call) ; ASR (answer seize ratio) ; CIC (Consecutive Incomplete Calls)
Condition:	Is less than
Value:	10 Input the value for the alert
Alert condition add on:	Same day
Status:	Active
Email to send alarm:	admin@cdr-stats.com

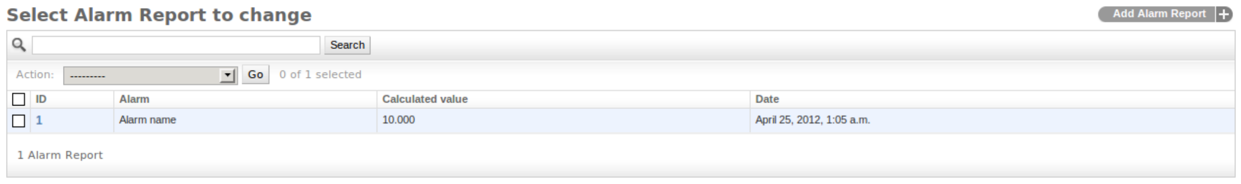
Save and add another | Save and continue editing | Save

Alarm-report

The alarmreport will be displayed from the following URL.

URL:

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



Select Alarm Report to change

Add Alarm Report

Search

Action: --------- Go 0 of 1 selected

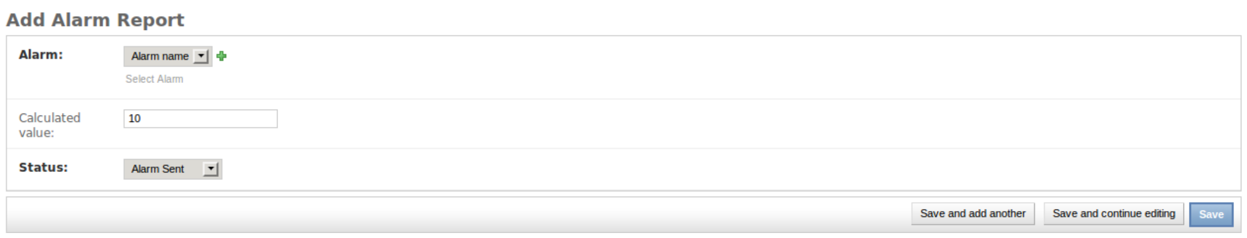
ID	Alarm	Calculated value	Date
1	Alarm name	10.000	April 25, 2012, 1:05 a.m.

1 Alarm Report

To Add/Update alarmreport

URL:

- http://localhost:8000/admin/cdr_alert/alarmreport/add/
- http://localhost:8000/admin/cdr_alert/alarmreport/1/


Add Alarm Report

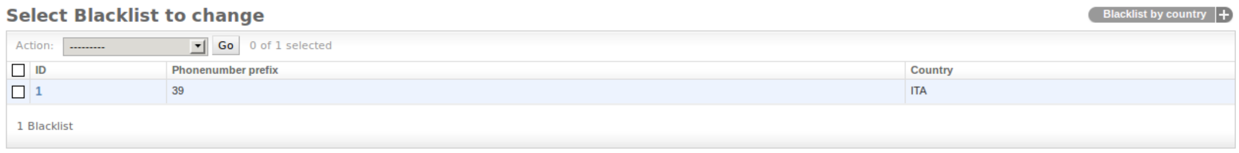
Alarm:	Alarm name (Select Alarm)
Calculated value:	10
Status:	Alarm Sent

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.

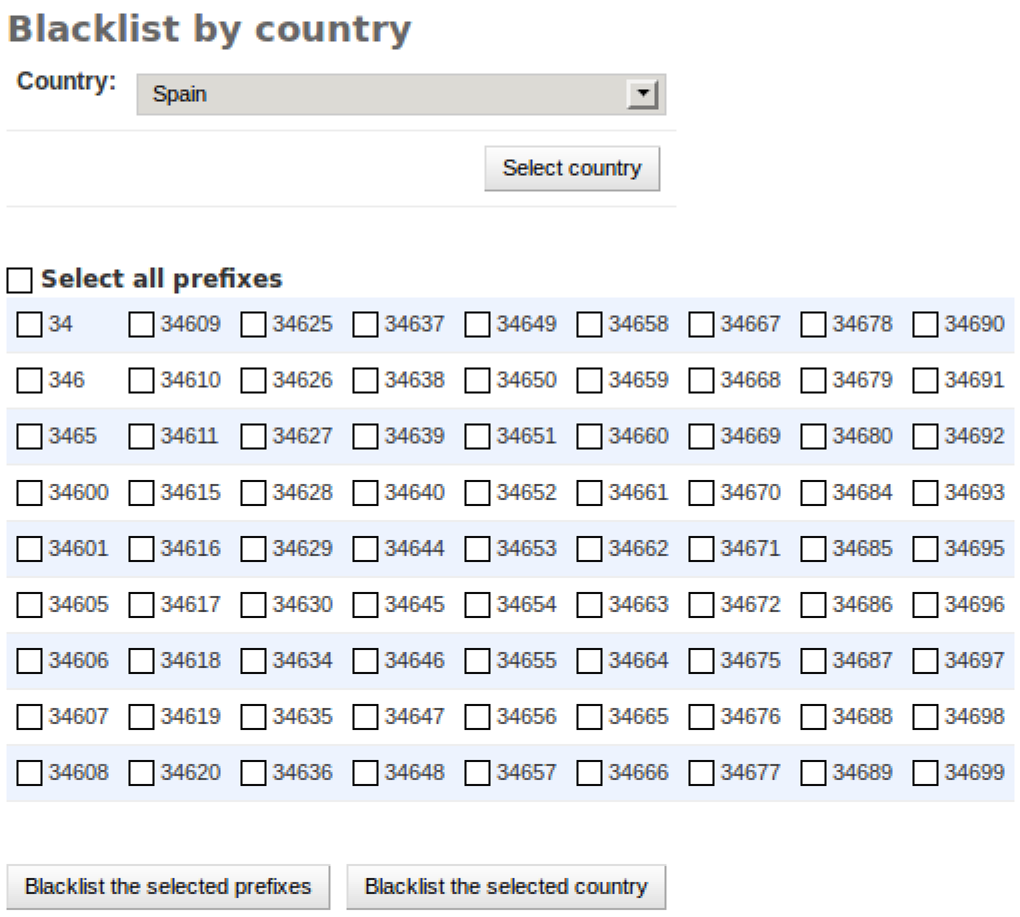
URL:

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


Select Blacklist to change

ID	Phonenumber prefix	Country
1	39	ITA

1 Blacklist


Blacklist by country

Country: Spain

34	34609	34625	34637	34649	34658	34667	34678	34690
346	34610	34626	34638	34650	34659	34668	34679	34691
3465	34611	34627	34639	34651	34660	34669	34680	34692
34600	34615	34628	34640	34652	34661	34670	34684	34693
34601	34616	34629	34644	34653	34662	34671	34685	34695
34605	34617	34630	34645	34654	34663	34672	34686	34696
34606	34618	34634	34646	34655	34664	34675	34687	34697
34607	34619	34635	34647	34656	34665	34676	34688	34698
34608	34620	34636	34648	34657	34666	34677	34689	34699

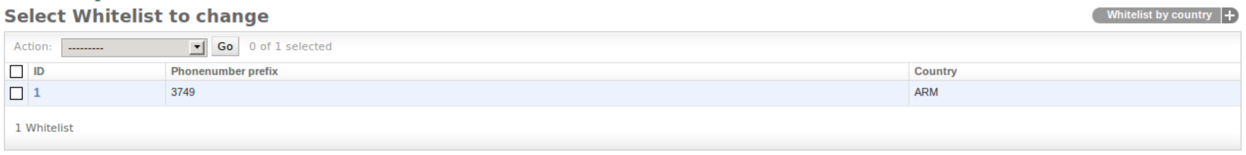
Blacklist the selected prefixes
Blacklist the selected country

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.

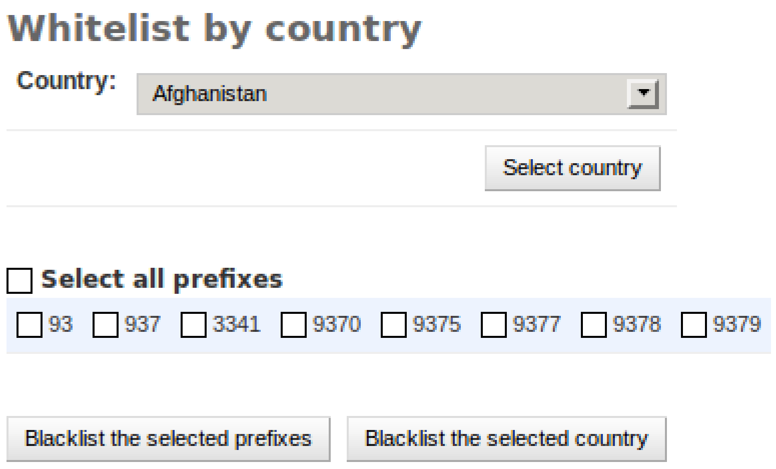
URL:

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


Select Whitelist to change
Whitelist by country
Action: --------- Go 0 of 1 selected

ID	Phonenumber prefix	Country
1	3749	ARM

1 Whitelist

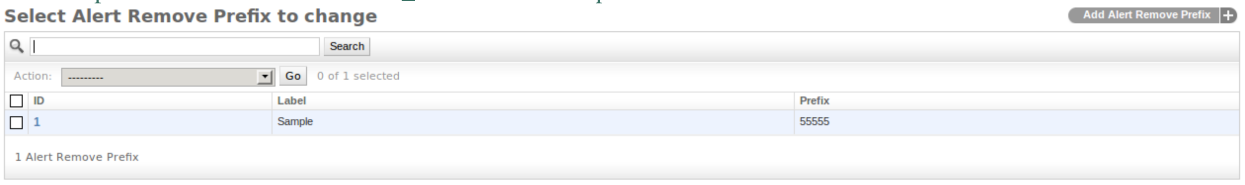


Alert-remove-prefix

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/


Select Alert Remove Prefix to change
Add Alert Remove Prefix
Search
Action: --------- Go 0 of 1 selected

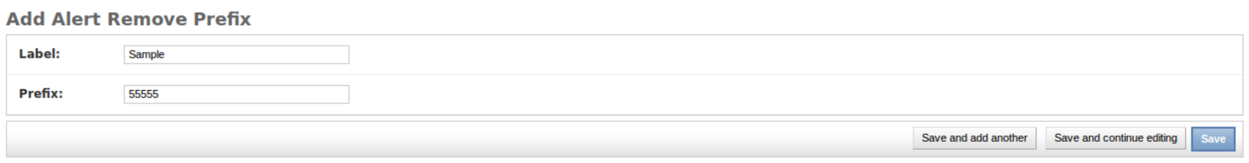
ID	Label	Prefix
1	Sample	55555

1 Alert Remove Prefix

To Add/Update alert-remove prefix

URL:

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



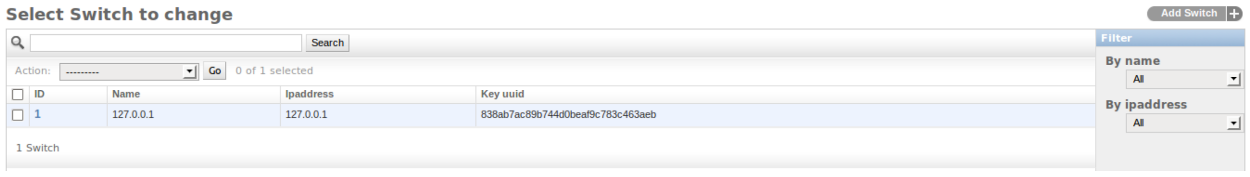
Add Alert Remove Prefix

Label:	Sample
Prefix:	55555

Switch

URL:

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



Select Switch to change

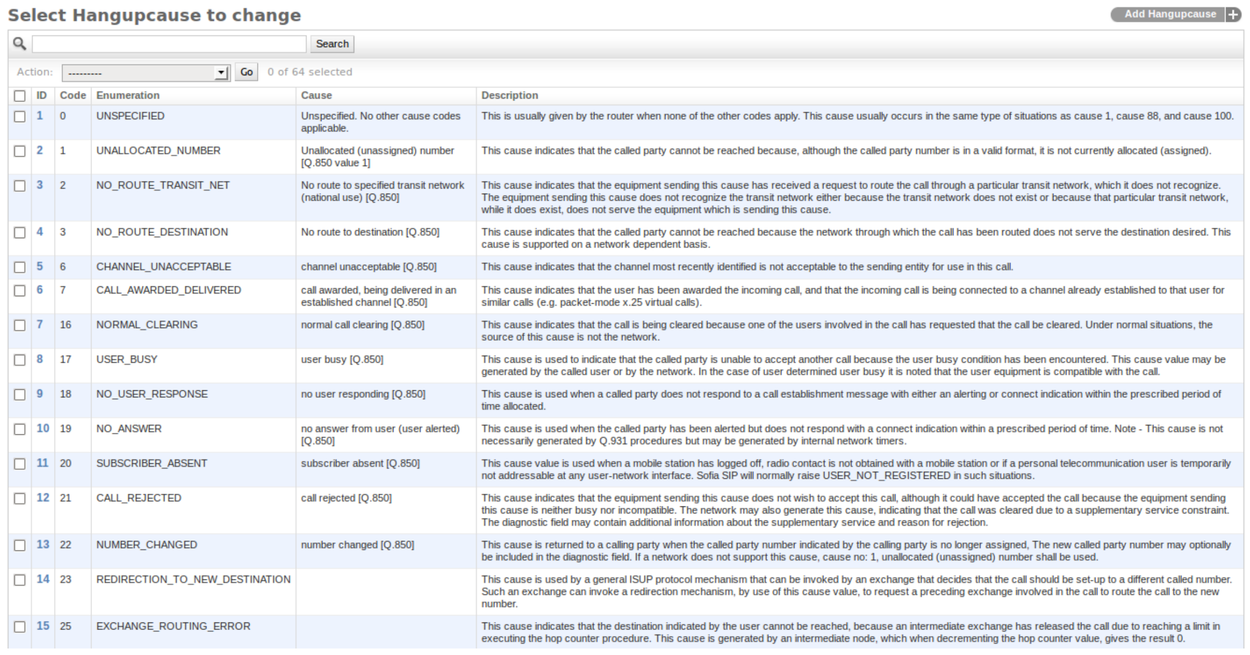
ID	Name	Ipaddress	Key uuid
1	127.0.0.1	127.0.0.1	838ab7ac89b744d0beaf9c783c463aeb

1 Switch

HangupCause

URL:

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



Select Hangupcause to change

ID	Code	Enumeration	Cause	Description
1	0	UNSPECIFIED	Unspecified. No other cause codes applicable.	This is usually given by the router when none of the other codes apply. This cause usually occurs in the same type of situations as cause 1, cause 88, and cause 100.
2	1	UNALLOCATED_NUMBER	Unallocated (unassigned) number [Q.850 value 1]	This cause indicates that the called party cannot be reached because, although the called party number is in a valid format, it is not currently allocated (assigned).
3	2	NO_ROUTE_TRANSIT_NET	No route to specified transit network (national use) [Q.850]	This cause indicates that the equipment sending this cause has received a request to route the call through a particular transit network, which it does not recognize. The equipment sending this cause does not recognize the transit network either because the transit network does not exist or because that particular transit network, while it does exist, does not serve the equipment which is sending this cause.
4	3	NO_ROUTE_DESTINATION	No route to destination [Q.850]	This cause indicates that the called party cannot be reached because the network through which the call has been routed does not serve the destination desired. This cause is supported on a network dependent basis.
5	6	CHANNEL_UNACCEPTABLE	channel unacceptable [Q.850]	This cause indicates that the channel most recently identified is not acceptable to the sending entity for use in this call.
6	7	CALL_AWARDED_DELIVERED	call awarded, being delivered in an established channel [Q.850]	This cause indicates that the user has been awarded the incoming call, and that the incoming call is being connected to a channel already established to that user for similar calls (e.g. packet-mode x.25 virtual calls).
7	16	NORMAL_CLEARING	normal call clearing [Q.850]	This cause indicates that the call is being cleared because one of the users involved in the call has requested that the call be cleared. Under normal situations, the source of this cause is not the network.
8	17	USER_BUSY	user busy [Q.850]	This cause is used to indicate that the called party is unable to accept another call because the user busy condition has been encountered. This cause value may be generated by the called user or by the network. In the case of user determined user busy it is noted that the user equipment is compatible with the call.
9	18	NO_USER_RESPONSE	no user responding [Q.850]	This cause is used when a called party does not respond to a call establishment message with either an alerting or connect indication within the prescribed period of time allocated.
10	19	NO_ANSWER	no answer from user (user alerted) [Q.850]	This cause is used when the called party has been alerted but does not respond with a connect indication within a prescribed period of time. Note - This cause is not necessarily generated by Q.931 procedures but may be generated by internal network timers.
11	20	SUBSCRIBER_ABSENT	subscriber absent [Q.850]	This cause value is used when a mobile station has logged off, radio contact is not obtained with a mobile station or if a personal telecommunication user is temporarily not addressable at any user-network interface. Sofia SIP will normally raise USER_NOT_REGISTERED in such situations.
12	21	CALL_REJECTED	call rejected [Q.850]	This cause indicates that the equipment sending this cause does not wish to accept this call, although it could have accepted the call because the equipment sending this cause is neither busy nor incompatible. The network may also generate this cause, indicating that the call was cleared due to a supplementary service constraint. The diagnostic field may contain additional information about the supplementary service and reason for rejection.
13	22	NUMBER_CHANGED	number changed [Q.850]	This cause is returned to a calling party when the called party number indicated by the calling party is no longer assigned, The new called party number may optionally be included in the diagnostic field. If a network does not support this cause, cause no: 1, unallocated (unassigned) number shall be used.
14	23	REDIRECTION_TO_NEW_DESTINATION		This cause is used by a general ISUP protocol mechanism that can be invoked by an exchange that decides that the call should be set-up to a different called number. Such an exchange can invoke a redirection mechanism, by use of this cause value, to request a preceding exchange involved in the call to route the call to the new number.
15	25	EXCHANGE_ROUTING_ERROR		This cause indicates that the destination indicated by the user cannot be reached, because an intermediate exchange has released the call due to reaching a limit in executing the hop counter procedure. This cause is generated by an intermediate node, which when decrementing the hop counter value, gives the result 0.

5.4 Customer Panel

User Interface :

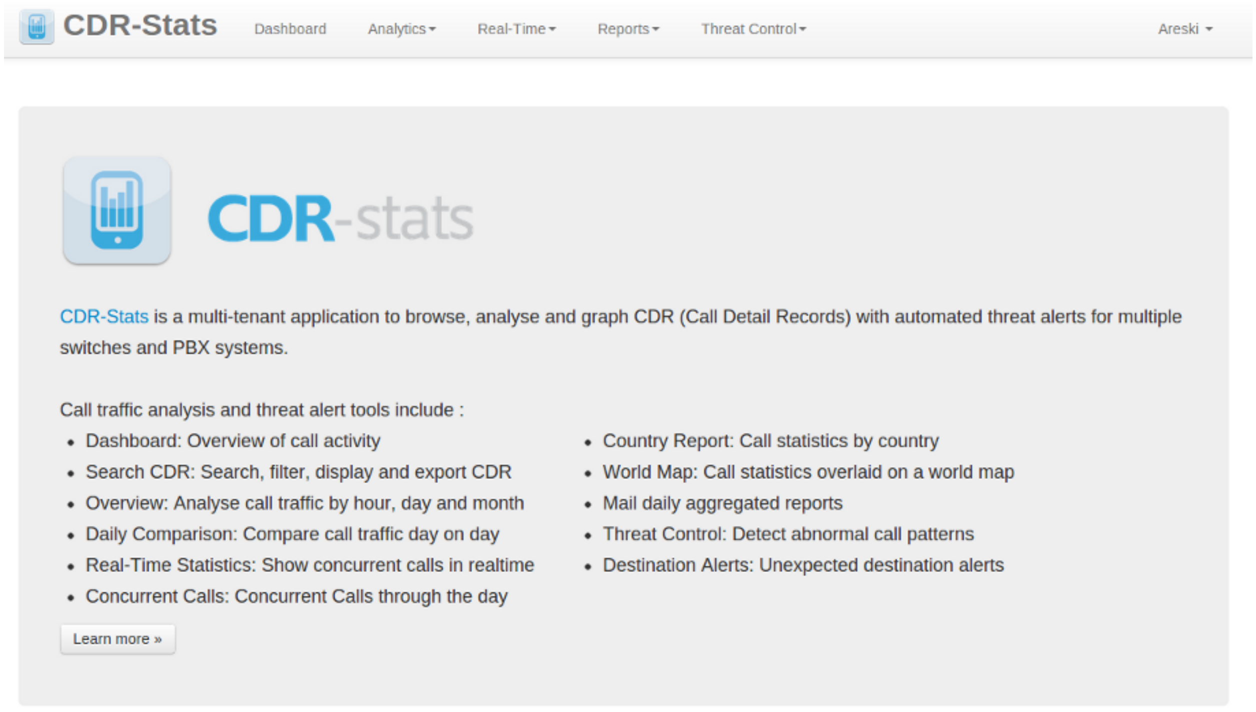
This application provides a user interface...

http://localhost:8000/

5.4.1 Screenshot with Features

Index

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



CDR-Stats is a multi-tenant application to browse, analyse and graph CDR (Call Detail Records) with automated threat alerts for multiple switches and PBX systems.

Call traffic analysis and threat alert tools include :

- 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
- Concurrent Calls: Concurrent Calls through the day
- 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

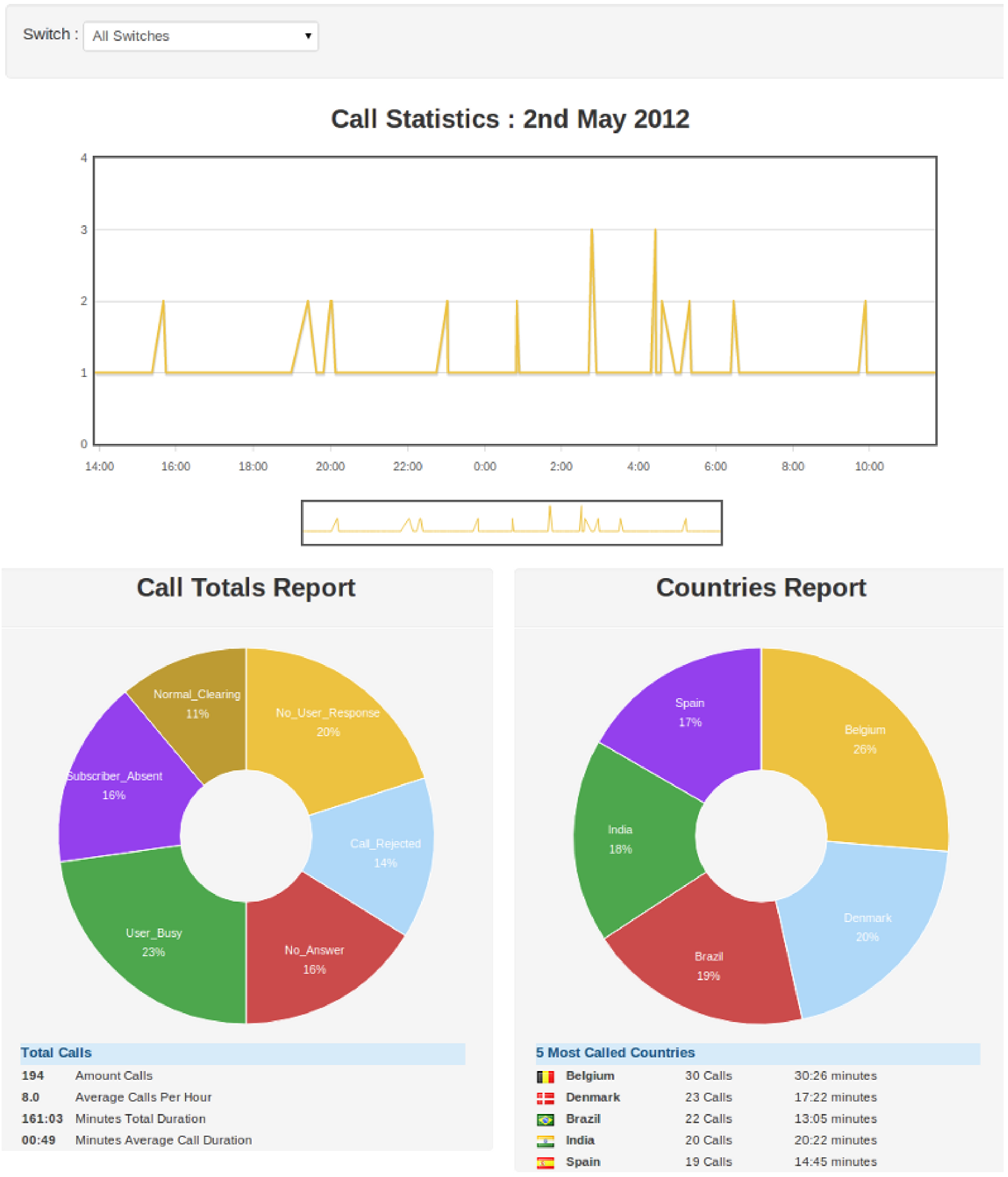
Learn more »

Dashboard

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

URL:

- http://localhost:8000/dashboard/



Total Calls

194	Amount Calls
8.0	Average Calls Per Hour
161:03	Minutes Total Duration
00:49	Minutes Average Call Duration

5 Most Called Countries

Belgium	30 Calls	30:26 minutes
Denmark	23 Calls	17:22 minutes
Brazil	22 Calls	13:05 minutes
India	20 Calls	20:22 minutes
Spain	19 Calls	14:45 minutes

CDR-View

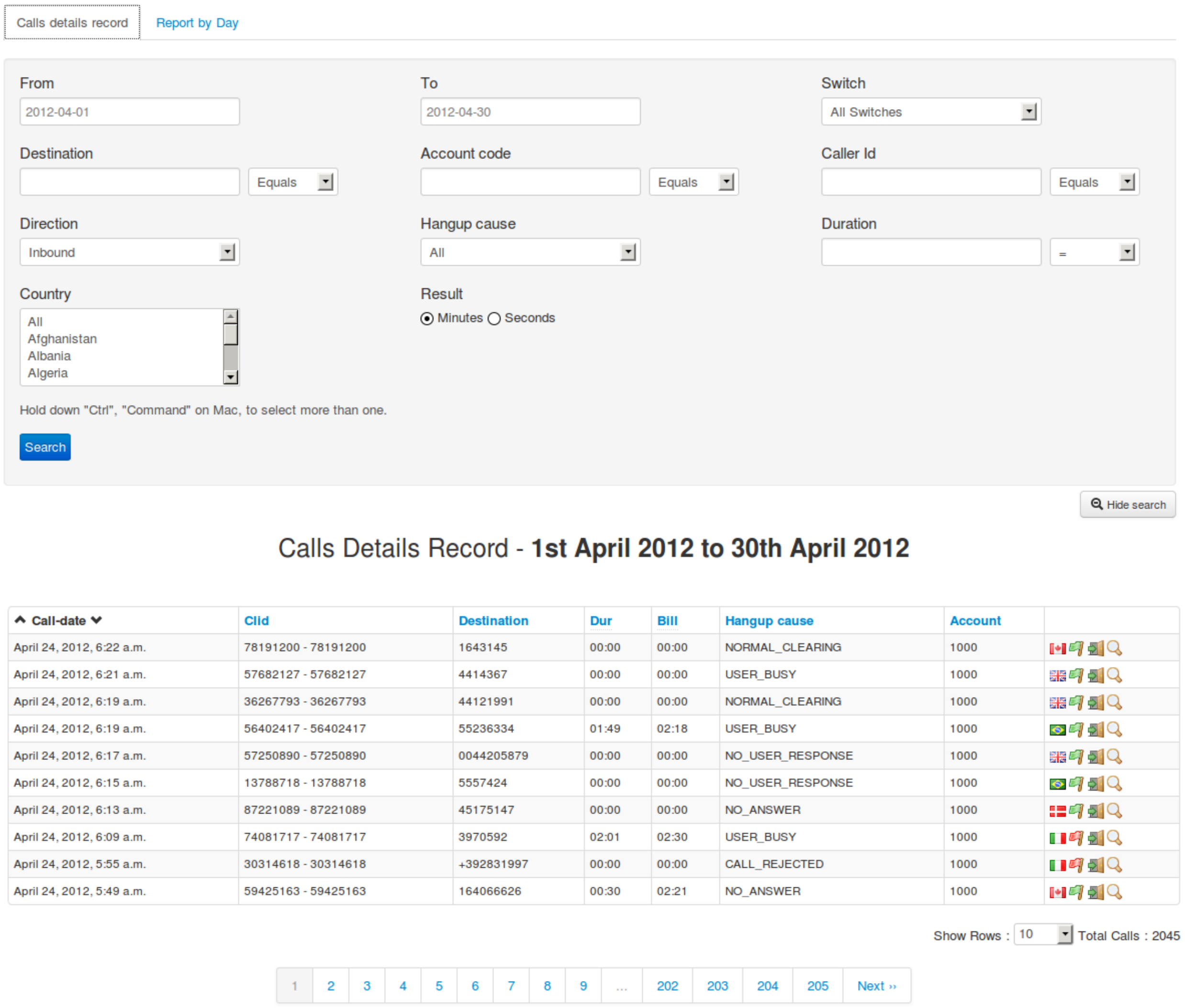
Call detail records listed in table format which can be exported to CSV file.

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 | Report by Day

From: 2012-04-01
To: 2012-04-30
Switch: All Switches

Destination: Equals
Account code: Equals
Caller Id: Equals

Direction: Inbound
Hangup cause: All
Duration: =

Country: All, Afghanistan, Albania, Algeria

Result: [✓ Minutes] Seconds

Hold down "Ctrl", "Command" on Mac, to select more than one.

Search

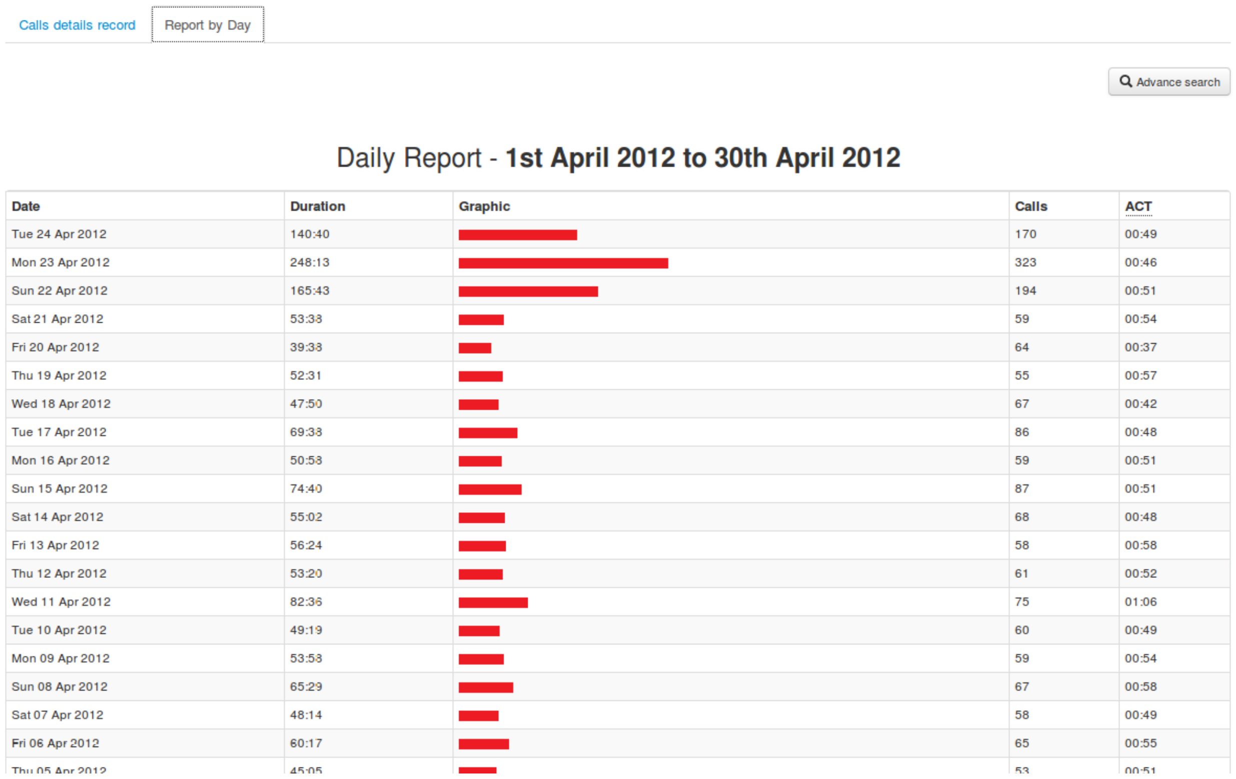
Hide search

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

Call-date	Clid	Destination	Dur	Bill	Hangup cause	Account
April 24, 2012, 6:22 a.m.	78191200 - 78191200	1643145	00:00	00:00	NORMAL_CLEARING	1000
April 24, 2012, 6:21 a.m.	57682127 - 57682127	4414367	00:00	00:00	USER_BUSY	1000
April 24, 2012, 6:19 a.m.	36267793 - 36267793	44121991	00:00	00:00	NORMAL_CLEARING	1000
April 24, 2012, 6:19 a.m.	56402417 - 56402417	55236334	01:49	02:18	USER_BUSY	1000
April 24, 2012, 6:17 a.m.	57250890 - 57250890	0044205879	00:00	00:00	NO_USER_RESPONSE	1000
April 24, 2012, 6:15 a.m.	13788718 - 13788718	5557424	00:00	00:00	NO_USER_RESPONSE	1000
April 24, 2012, 6:13 a.m.	87221089 - 87221089	45175147	00:00	00:00	NO_ANSWER	1000
April 24, 2012, 6:09 a.m.	74081717 - 74081717	3970592	02:01	02:30	USER_BUSY	1000
April 24, 2012, 5:55 a.m.	30314618 - 30314618	+392831997	00:00	00:00	CALL_REJECTED	1000
April 24, 2012, 5:49 a.m.	59425163 - 59425163	164066626	00:30	02:21	NO_ANSWER	1000

Show Rows : 10 Total Calls : 2045

1 2 3 4 5 6 7 8 9 ... 202 203 204 205 Next ››



Daily Report - **1st April 2012 to 30th April 2012**

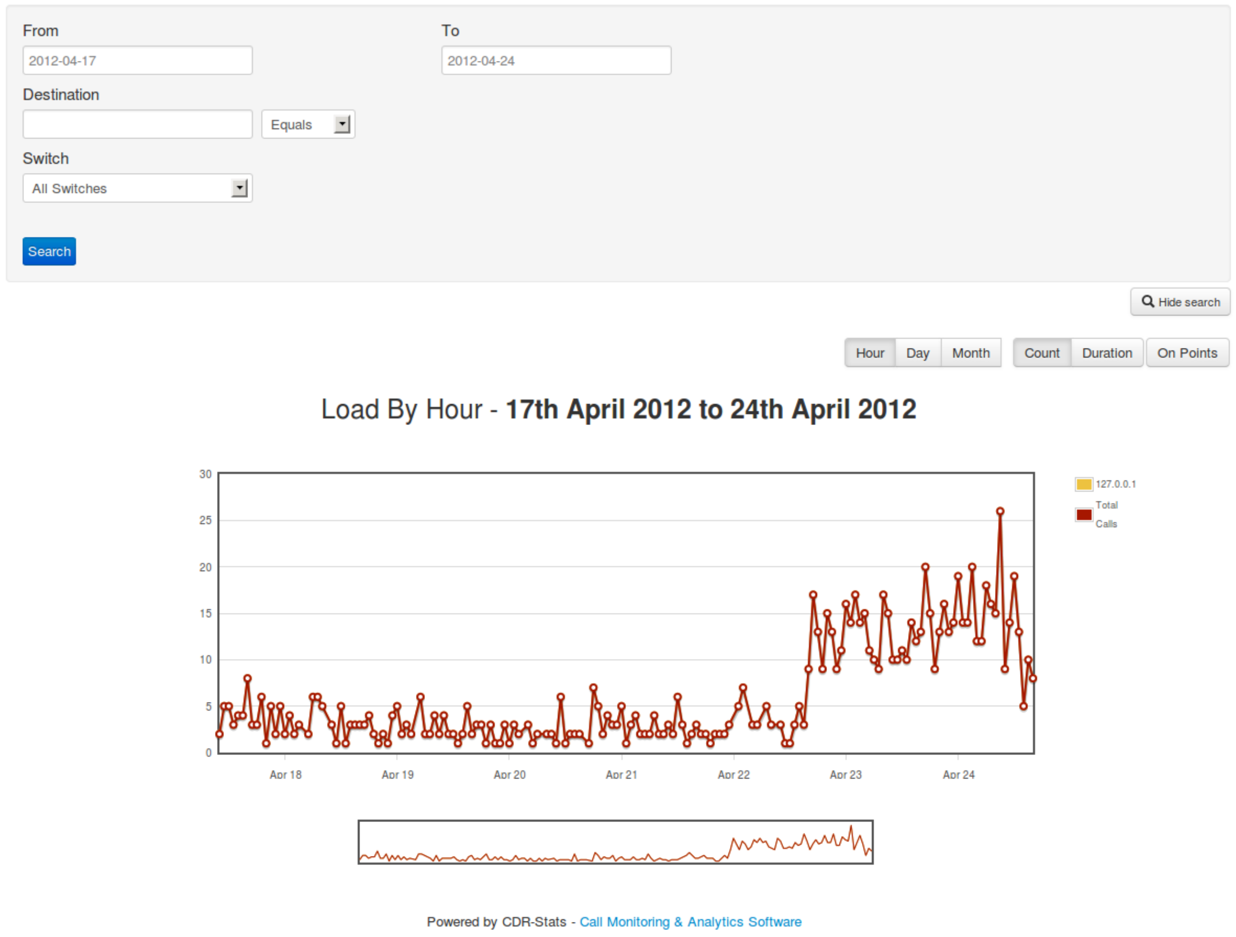
Date	Duration	Graphic	Calls	ACT
Tue 24 Apr 2012	140:40		170	00:49
Mon 23 Apr 2012	248:13		323	00:46
Sun 22 Apr 2012	165:43		194	00:51
Sat 21 Apr 2012	53:38		59	00:54
Fri 20 Apr 2012	39:38		64	00:37
Thu 19 Apr 2012	52:31		55	00:57
Wed 18 Apr 2012	47:50		67	00:42
Tue 17 Apr 2012	69:38		86	00:48
Mon 16 Apr 2012	50:58		59	00:51
Sun 15 Apr 2012	74:40		87	00:51
Sat 14 Apr 2012	55:02		68	00:48
Fri 13 Apr 2012	56:24		58	00:58
Thu 12 Apr 2012	53:20		61	00:52
Wed 11 Apr 2012	82:36		75	01:06
Tue 10 Apr 2012	49:19		60	00:49
Mon 09 Apr 2012	53:58		59	00:54
Sun 08 Apr 2012	65:29		67	00:58
Sat 07 Apr 2012	48:14		58	00:49
Fri 06 Apr 2012	60:17		65	00:55
Thu 05 Apr 2012	45:05		53	00:51

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/

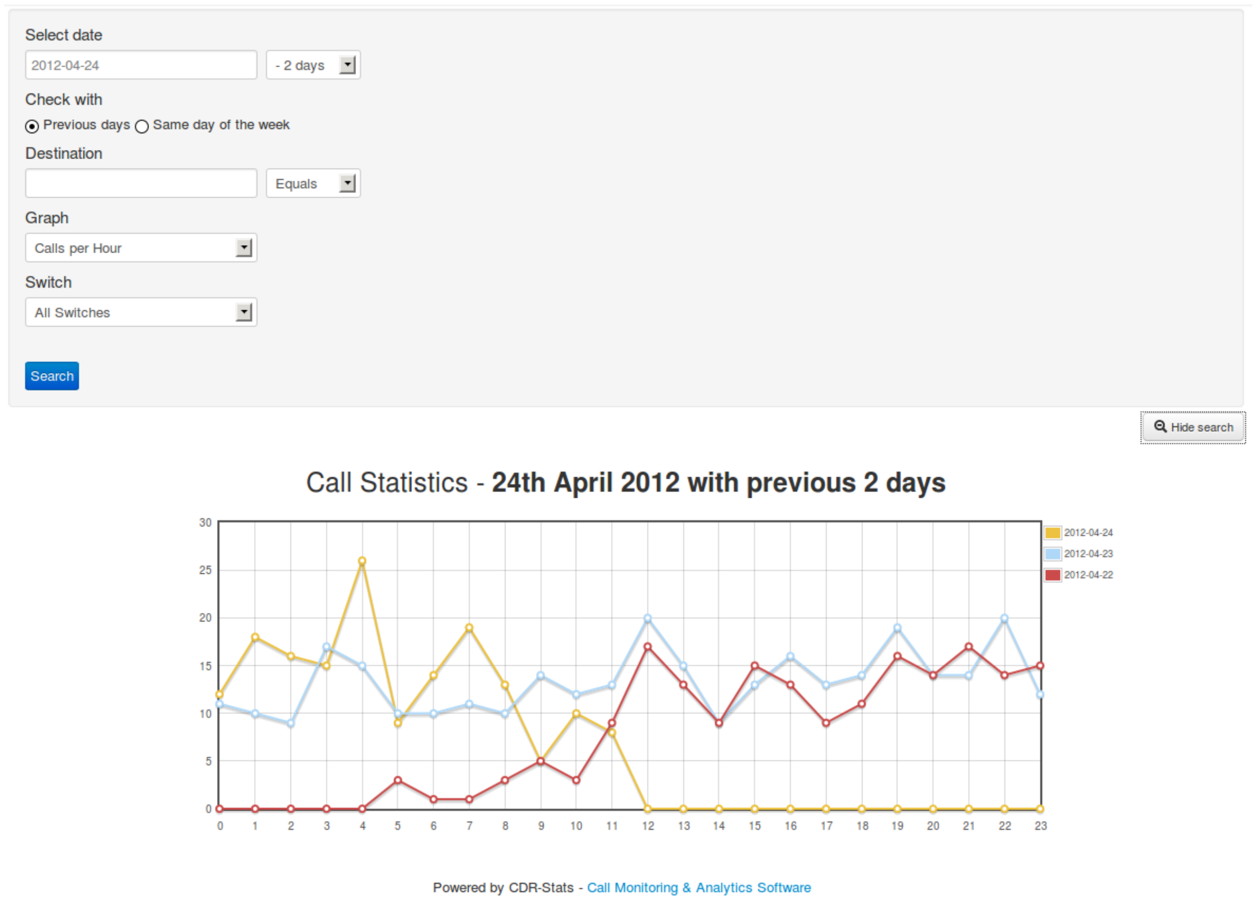

From: 2012-04-17
To: 2012-04-24
Destination: Equals
Switch: All Switches

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/


Select date

2012-04-24 | - 2 days

Check with

(● Previous days) ○ Same day of the week

Destination

Equals

Graph

Calls per Hour

Switch

All Switches

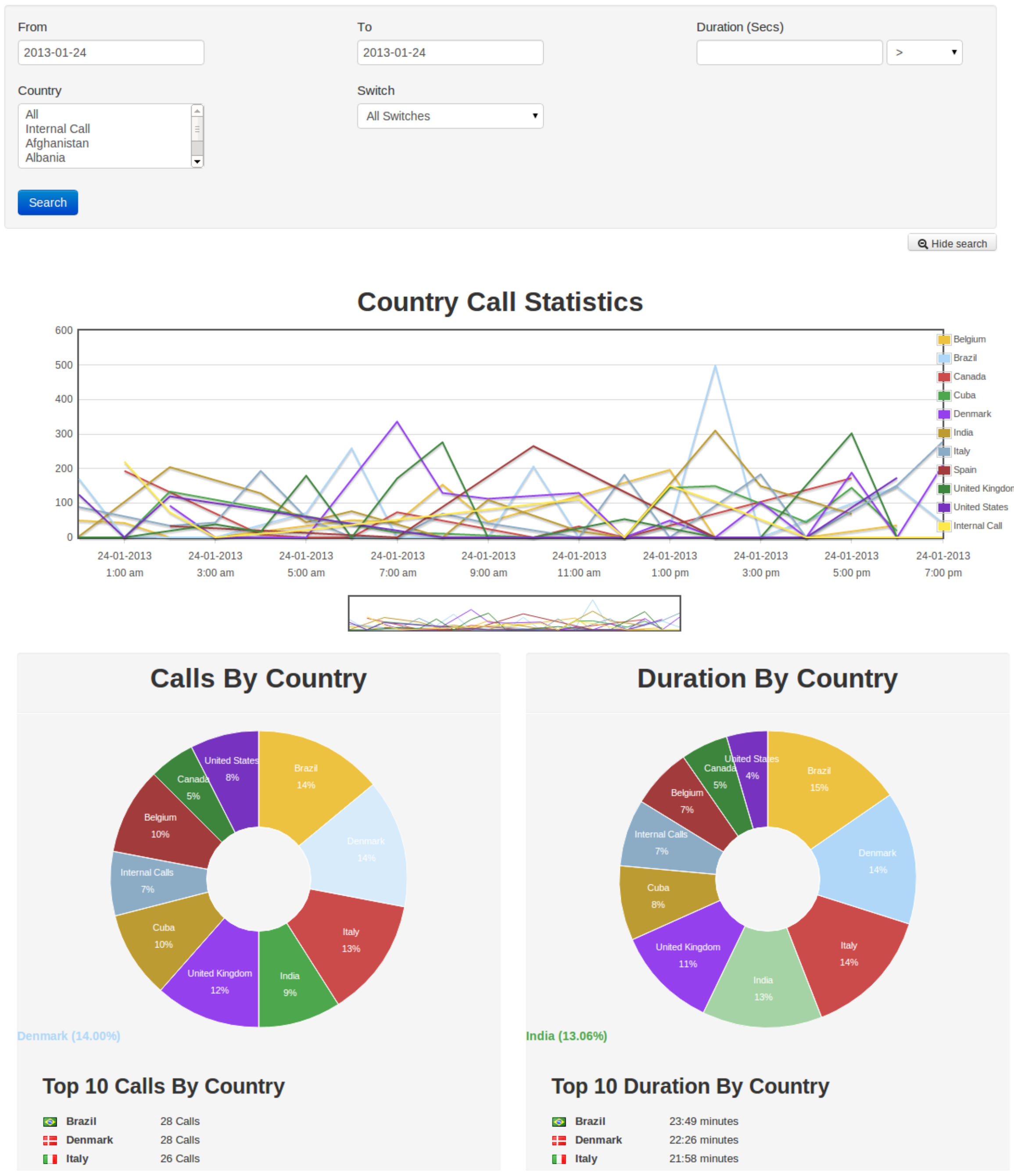
Powered by CDR-Stats - Call Monitoring & Analytics Software

CDR-Country-Report

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

URL:

- http://localhost:8000/country_report/



From: 2013-01-24

To: 2013-01-24

Duration (Secs): >

Country: All, Internal Call, Afghanistan, Albania

Switch: All Switches

Top 10 Calls By Country

Brazil	28 Calls
Denmark	28 Calls
Italy	26 Calls

Top 10 Duration By Country

Brazil	23:49 minutes
Denmark	22:26 minutes
Italy	21:58 minutes

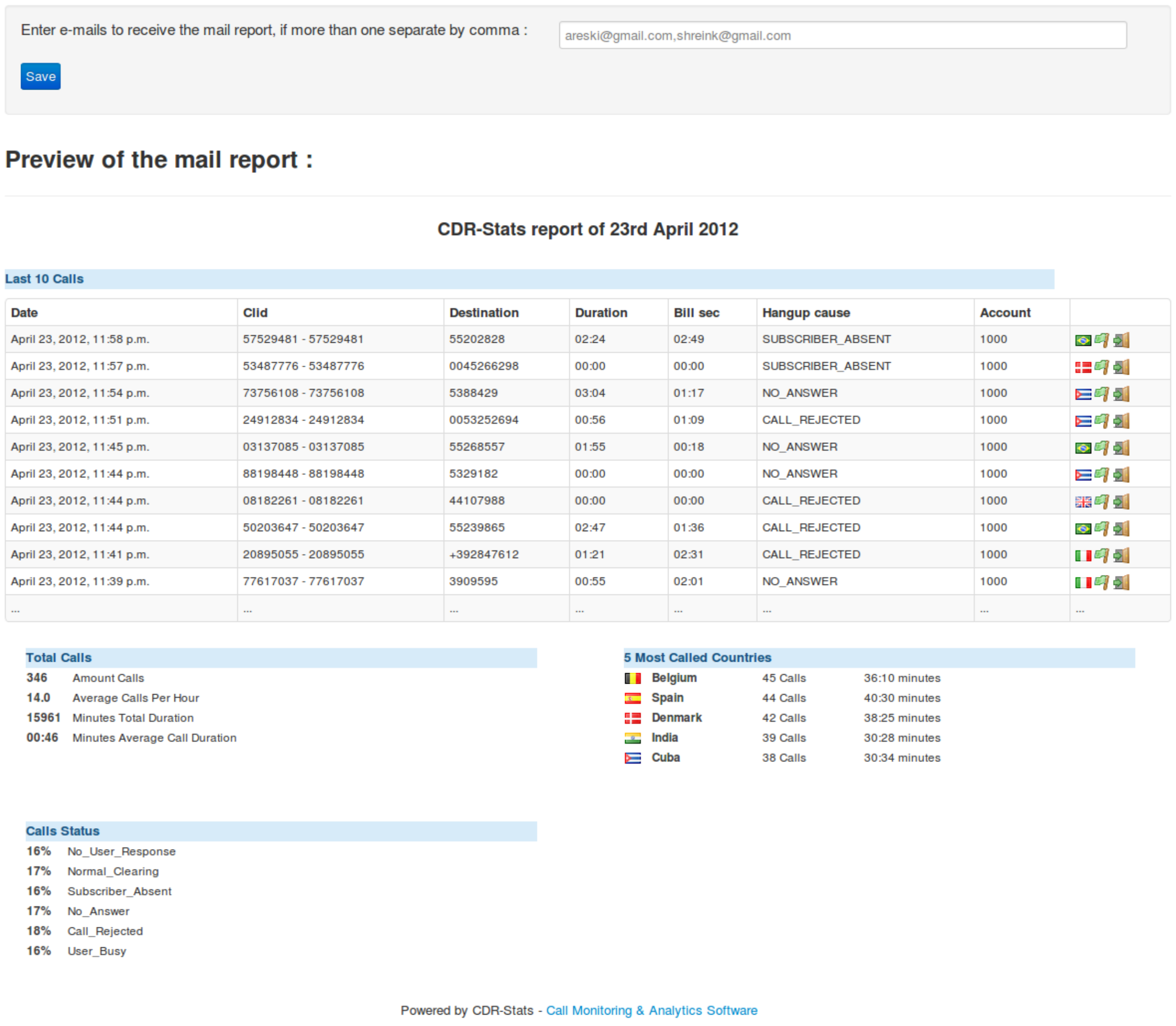
Mail-Report

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:

- http://localhost:8000/mail_report/



Enter e-mails to receive the mail report, if more than one separate by comma : areski@gmail.com,shreink@gmail.com

Preview of the mail report :

CDR-Stats report of 23rd April 2012

Last 10 Calls

Date	Clid	Destination	Duration	Bill sec	Hangup cause	Account	
April 23, 2012, 11:58 p.m.	57529481 - 57529481	55202828	02:24	02:49	SUBSCRIBER_ABSENT	1000	
April 23, 2012, 11:57 p.m.	53487776 - 53487776	0045266298	00:00	00:00	SUBSCRIBER_ABSENT	1000	
April 23, 2012, 11:54 p.m.	73756108 - 73756108	5388429	03:04	01:17	NO_ANSWER	1000	
April 23, 2012, 11:51 p.m.	24912834 - 24912834	0053252694	00:56	01:09	CALL_REJECTED	1000	
April 23, 2012, 11:45 p.m.	03137085 - 03137085	55268557	01:55	00:18	NO_ANSWER	1000	
April 23, 2012, 11:44 p.m.	88198448 - 88198448	5329182	00:00	00:00	NO_ANSWER	1000	
April 23, 2012, 11:44 p.m.	08182261 - 08182261	44107988	00:00	00:00	CALL_REJECTED	1000	
April 23, 2012, 11:44 p.m.	50203647 - 50203647	55239865	02:47	01:36	CALL_REJECTED	1000	
April 23, 2012, 11:41 p.m.	20895055 - 20895055	+392847612	01:21	02:31	CALL_REJECTED	1000	
April 23, 2012, 11:39 p.m.	77617037 - 77617037	3909595	00:55	02:01	NO_ANSWER	1000	
...	...	...	...	...	...	...	...

Total Calls

346	Amount Calls
14.0	Average Calls Per Hour
15961	Minutes Total Duration
00:46	Minutes Average Call Duration

5 Most Called Countries

Belgium	45 Calls	36:10 minutes
Spain	44 Calls	40:30 minutes
Denmark	42 Calls	38:25 minutes
India	39 Calls	30:28 minutes
Cuba	38 Calls	30:34 minutes

Calls Status

16%	No_User_Response
17%	Normal_Clearing
16%	Subscriber_Absent
17%	No_Answer
18%	Call_Rejected
16%	User_Busy

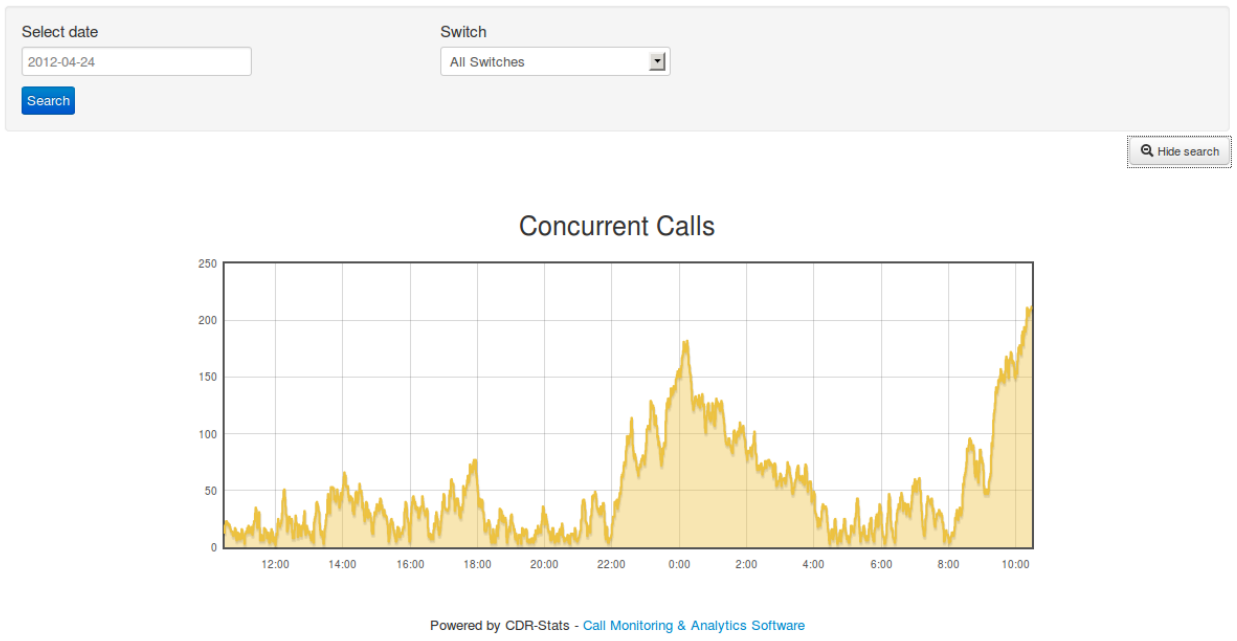
Powered by CDR-Stats - Call Monitoring & Analytics Software

Concurrent-call-report

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

URL:

- http://localhost:8000/cdr_concurrent_calls/

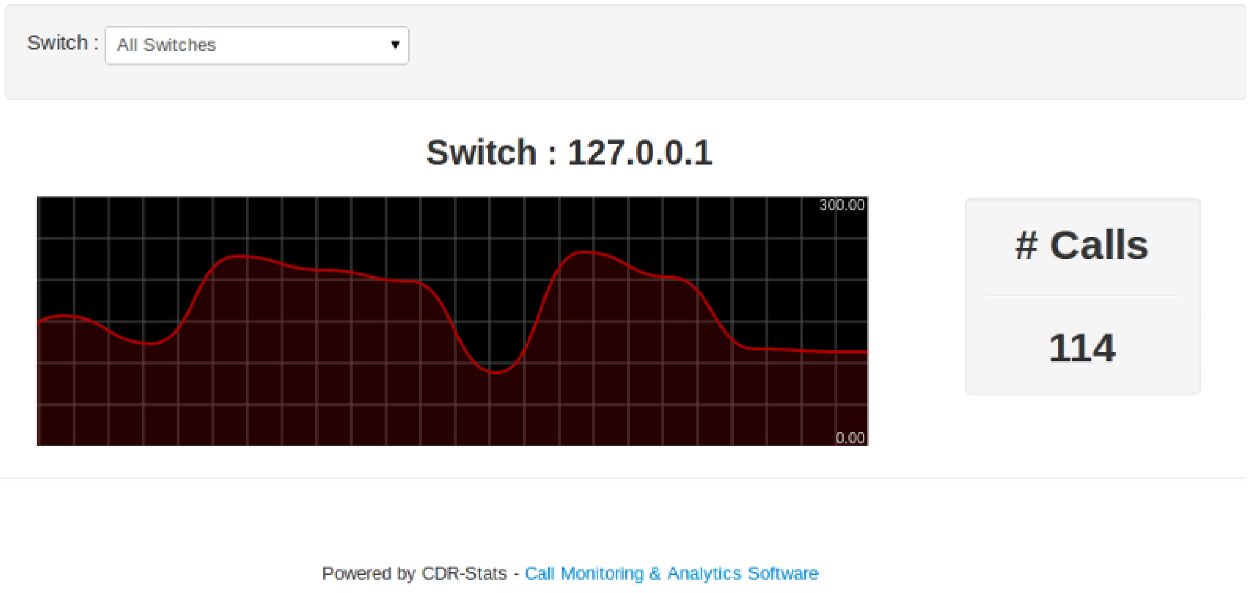


Realtime-Report

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

URL:

- http://localhost:8000/cdr_realtime/

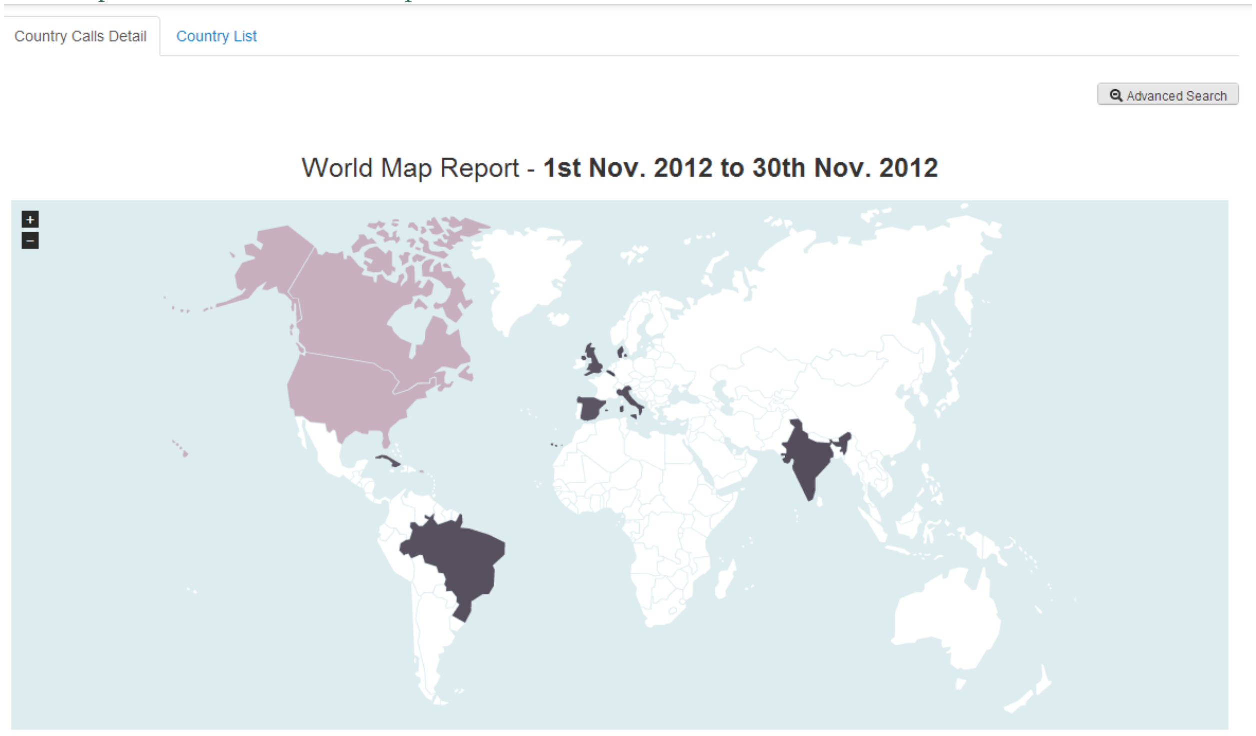


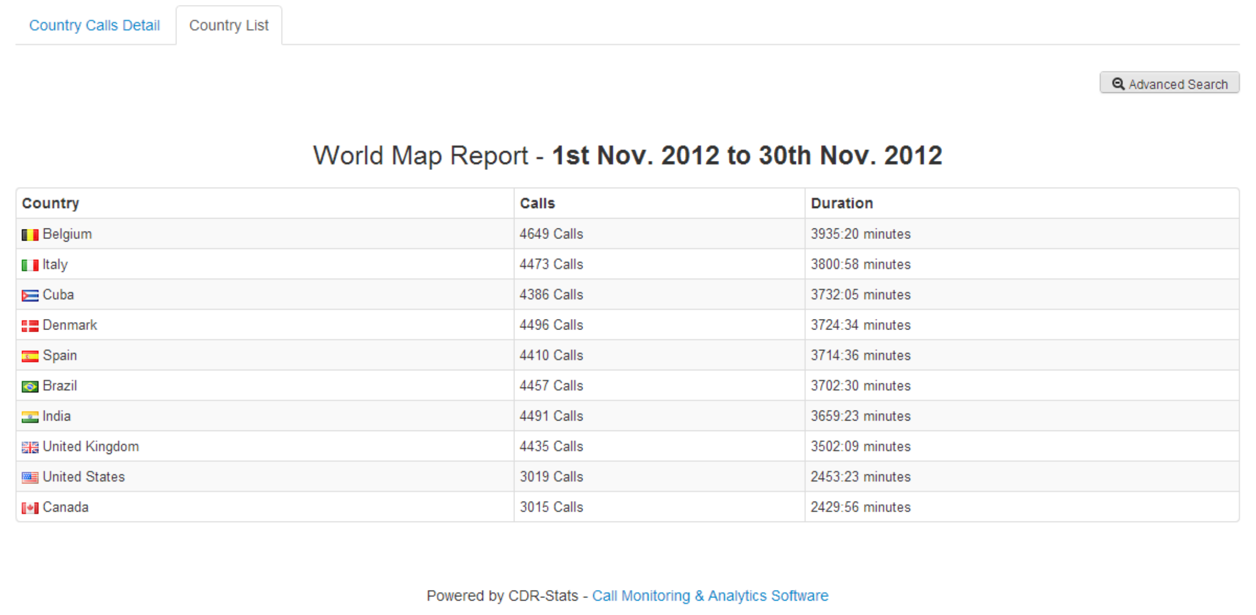
World Map Report

A distriibution 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**

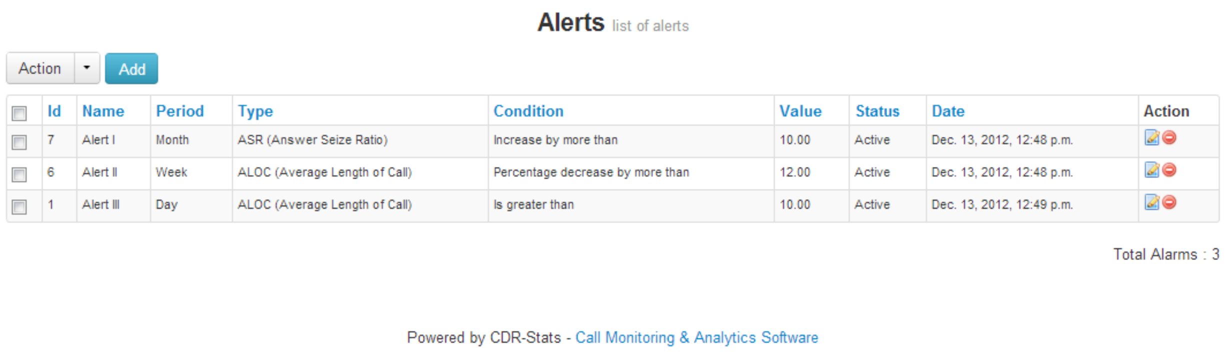
Country	Calls	Duration
Belgium	4649 Calls	3935:20 minutes
Italy	4473 Calls	3800:58 minutes
Cuba	4386 Calls	3732:05 minutes
Denmark	4496 Calls	3724:34 minutes
Spain	4410 Calls	3714:36 minutes
Brazil	4457 Calls	3702:30 minutes
India	4491 Calls	3659:23 minutes
United Kingdom	4435 Calls	3502:09 minutes
United States	3019 Calls	2453:23 minutes
Canada	3015 Calls	2429:56 minutes

Powered by CDR-Stats - Call Monitoring & Analytics Software

Alert Settings

URL:

- http://localhost:8000/alert/



Alerts list of alerts

Action | Add

	Id	Name	Period	Type	Condition	Value	Status	Date	Action
☐	7	Alert I	Month	ASR (Answer Seize Ratio)	Increase by more than	10.00	Active	Dec. 13, 2012, 12:48 p.m.	
☐	6	Alert II	Week	ALOC (Average Length of Call)	Percentage decrease by more than	12.00	Active	Dec. 13, 2012, 12:48 p.m.	
☐	1	Alert III	Day	ALOC (Average Length of Call)	Is greater than	10.00	Active	Dec. 13, 2012, 12:49 p.m.	

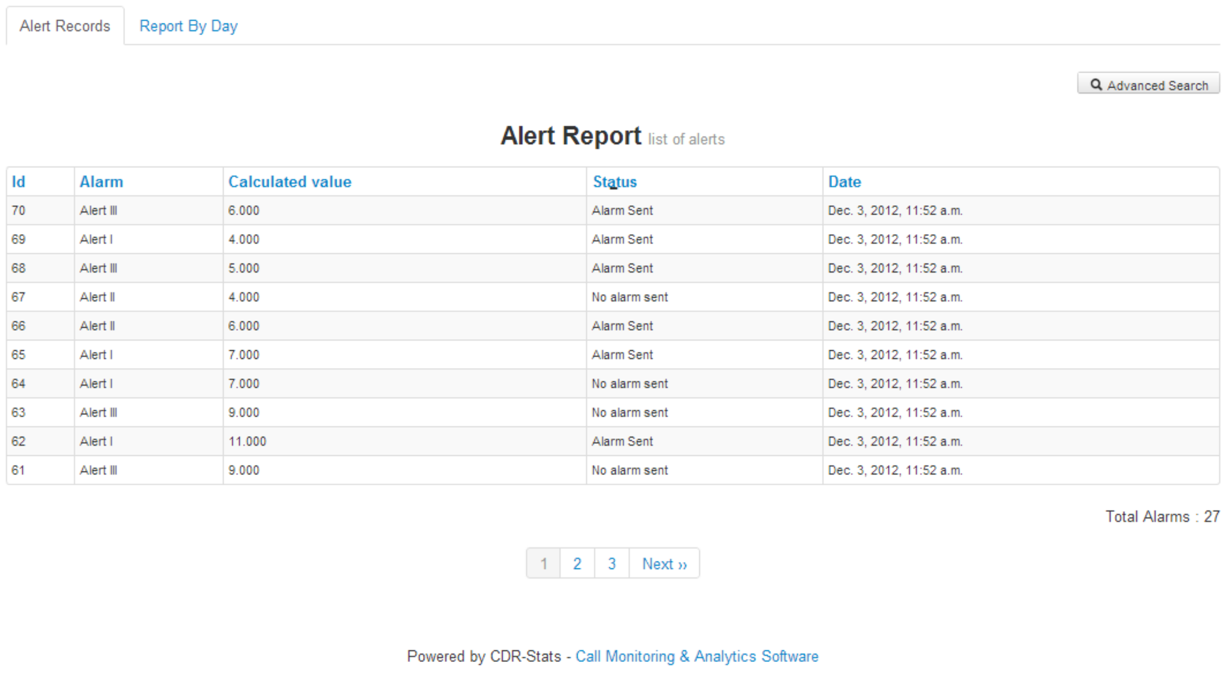
Total Alarms : 3

Powered by CDR-Stats - Call Monitoring & Analytics Software

Alert Report

URL:

- http://localhost:8000/alert_report/



Alert Records | Report By Day

Advanced Search

Alert Report list of alerts

Id	Alarm	Calculated value	Status	Date
70	Alert III	6.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
69	Alert I	4.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
68	Alert III	5.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
67	Alert II	4.000	No alarm sent	Dec. 3, 2012, 11:52 a.m.
66	Alert II	6.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
65	Alert I	7.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
64	Alert I	7.000	No alarm sent	Dec. 3, 2012, 11:52 a.m.
63	Alert III	9.000	No alarm sent	Dec. 3, 2012, 11:52 a.m.
62	Alert I	11.000	Alarm Sent	Dec. 3, 2012, 11:52 a.m.
61	Alert III	9.000	No alarm sent	Dec. 3, 2012, 11:52 a.m.

Total Alarms : 27

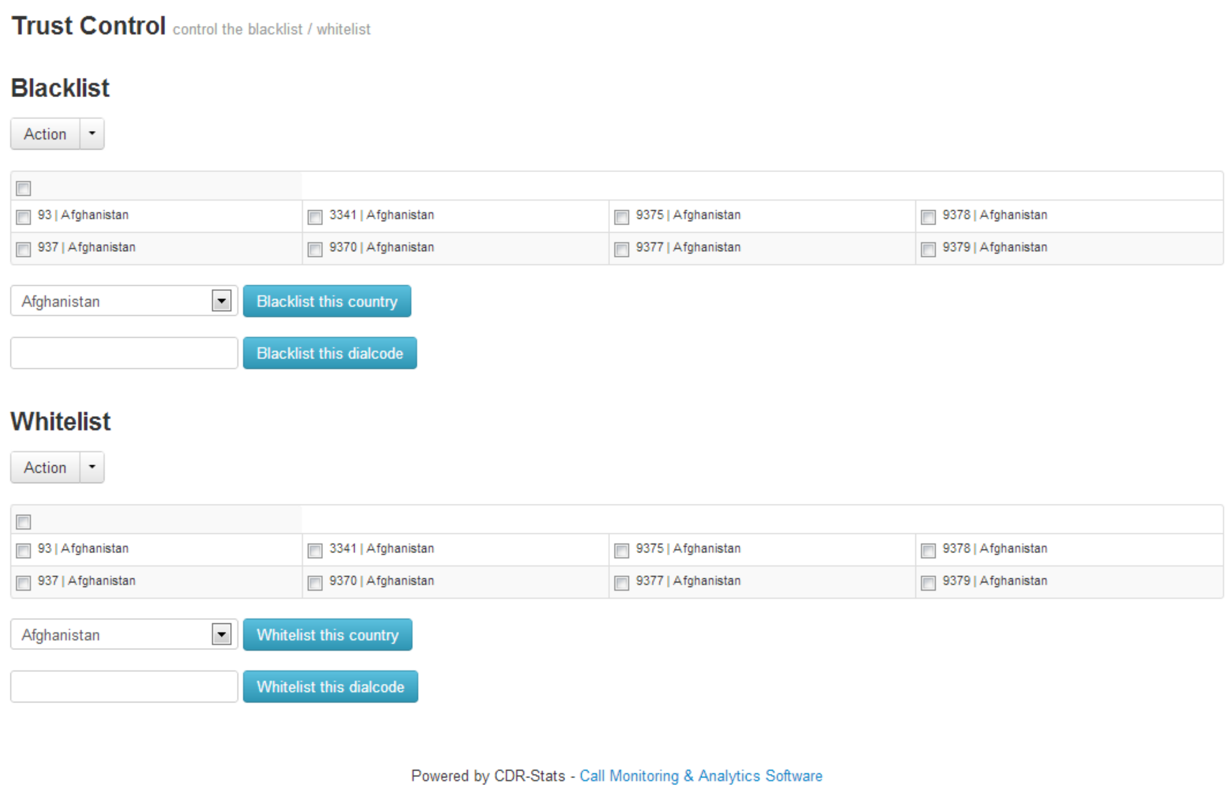
1 | 2 | 3 | Next ››

Powered by CDR-Stats - Call Monitoring & Analytics Software

Destination Control

URL:

- http://localhost:8000/trust_control/



Trust Control control the blacklist / whitelist

Blacklist

Action

93 | Afghanistan	3341 | Afghanistan	9375 | Afghanistan	9378 | Afghanistan
937 | Afghanistan	9370 | Afghanistan	9377 | Afghanistan	9379 | Afghanistan

Afghanistan — Blacklist this country

Blacklist this dialcode

Whitelist

Action

93 | Afghanistan	3341 | Afghanistan	9375 | Afghanistan	9378 | Afghanistan
937 | Afghanistan	9370 | Afghanistan	9377 | Afghanistan	9379 | Afghanistan

Afghanistan — Whitelist this country

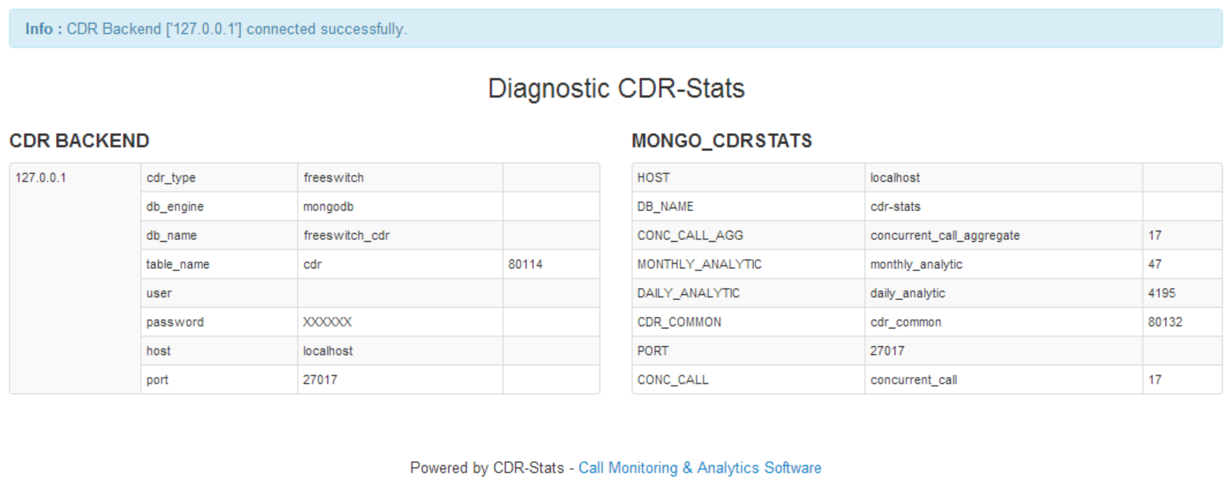
Whitelist this dialcode

Powered by CDR-Stats - Call Monitoring & Analytics Software

Diagnostic CDR-Stats

URL:

- http://localhost:8000/diagnostic/



Info : CDR Backend ['127.0.0.1'] connected successfully.

Diagnostic CDR-Stats

CDR BACKEND

127.0.0.1	cdr_type	freeswitch	
	db_engine	mongodb	
	db_name	freeswitch_cdr	
	table_name	cdr	80114
	user		
	password	XXXXXX	
	host	localhost	
	port	27017	

MONGO_CDRSTATS

HOST	localhost	
DB_NAME	cdr-stats	
CONC_CALL_AGG	concurrent_call_aggregate	17
MONTHLY_ANALYTIC	monthly_analytic	47
DAILY_ANALYTIC	daily_analytic	4195
CDR_COMMON	cdr_common	80132
PORT	27017	
CONC_CALL	concurrent_call	17

Powered by CDR-Stats - Call Monitoring & Analytics Software

CHAPTER

SIX

MONGODB

Web http://www.mongodb.org/

Download http://www.mongodb.org/downloads/

–

MongoDB is a scalable, high-performance, document-oriented schemaless database, everything in MongoDB is a document. There is no notion of a rigid table structure composed of columns and types.

Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas. The goal of MongoDB is to bridge the gap between key-value stores (which are fast and scalable) and relational databases (which have rich functionality).

6.1 Why MongoDB

Why did we choose MongoDB and what are the benefits? To answer these questions, we should enumerate some of the major features of MongoDB.

Document-oriented:

- Documents (objects) map well to programming language data types
- Embedded documents and arrays reduce need for joins
- Dynamically-typed (schema-less) for easy schema evolution

High performance:

- No joins and embedding make reads and writes fast
- Indexes including indexing of keys from embedded documents and arrays

High availability:

- Replicating servers with automatic master failover

A more detailed list of everything provided by mongoDB can be found at http://www.mongodb.org/display/DOCS/Introduction

As MongoDB is a Document-oriented datastore, it had a potential to store a huge number of CDR's, Call Detail Record formats vary between Telecom Switch types. For these reasons a NoSQL database is a very good candidate for a CDR warehouse.

6.2 Datastore Architecture

The MongoDB aggregation framework provides a means to calculate aggregate values without having to use Map-reduce (http://www.mongodb.org/display/DOCS/MapReduce). For those familiar with SQL, the aggregation framework can be used to do the kind of thing that SQL does with group-by and distinct, as well as some simple forms of self-joins.

The aggregation framework also provides projection facilities that can be used to reshape data. This includes the ability to add computed fields, to create new virtual sub-objects, and to extract sub-fields and bring them to the top-level of results.

update() replaces the document matching criteria entirely with objNew.

Shell syntax for update(): db.collection.update(criteria, objNew, upsert, multi)

Arguments:

- **criteria** - query which selects the record to update
- **objNew** - updated object or $ operators (e.g., $inc) which manipulate the object
- **upsert** - if this should be an "upsert" operation; that is, if the record(s) do not exist, insert one. Upsert only inserts a single document.
- **multi** - indicates if all documents matching criteria should be updated rather than just one. Can be useful with the $ operators below.

Shell syntax for $inc: { $inc : { field : value } } Increments "field" by the number "value" if "field" is present in the object, otherwise sets "field" to the number "value". This can also be used to decrement by using a negative "value".

6.3 Pre-Aggregated Reports

If you collect a large amount of data and you want to have access to aggregated information and reports, then you need a method to aggregate these data into a usable form. Pre-aggregating your data will provide performance gains when you try to retrieve that aggregrate information in realtime.

MongoDB is an engine is used for collecting and processing events in real time for use in generating up to the minute or second reports.

The first step in the aggregation process is to aggregate event data into the finest required granularity. Then use this aggregation to generate the next least specific level granularity and this repeat process until you have generated all required views.

6.4 One Document Per Day

Consider the following example schema for a solution that stores in a single document all statistics of a page for one day:

```
{
    _id: "20101010/site-1/apache_pb.gif",
    metadata: {
        date: ISODate("2000-10-10T00:00:00Z"),
        site: "site-1",
        page: "/apache_pb.gif" },
    daily: 5468426,
    hourly: {
        "0": 227850,
        "1": 210231,
        ...
        "23": 20457 },
    minute: {
        "0": 3612,
        "1": 3241,
        ...
        "1439": 2819 }
}
```

This approach has a couple of advantages:

- For every request on the website, you only need to update one document.
- Reports for time periods within the day, for a single page require fetching a single document.

There are, however, significant issues with this approach. The most significant issue is that, as you `upsert` data into the hourly and monthly fields, the document grows. Although MongoDB will pad the space allocated to documents, it will need to reallocate these documents multiple times throughout the day, which impacts performance.

6.5 Separate Documents by Granularity Level

Pre-allocating documents is a reasonable design for storing intra-day data, but the model breaks down when displaying data over longer multi-day periods like months or quarters. In these cases, consider storing daily statistics in a single document as above, and then aggregate monthly data into a separate document.

This introduce a second set of upsert operations to the data collection and aggregation portion of your application but the gains reduction in disk seeks on the queries, should be worth the costs. Consider the following example schema:

Daily Statistics:

```
{
    _id: "20101010/site-1/apache_pb.gif",
    metadata: {
        date: ISODate("2000-10-10T00:00:00Z"),
        site: "site-1",
        page: "/apache_pb.gif" },
    hourly: {
        "0": 227850,
        "1": 210231,
        ...
        "23": 20457 },
    minute: {
        "0": {
            "0": 3612,
            "1": 3241,
            ...
            "59": 2130 },
        "1": {
            "0": ...,
        },
        ...
        "23": {
            "59": 2819 }
    }
}
```

Monthly Statistics:

```
{
    _id: "201010/site-1/apache_pb.gif",
    metadata: {
        date: ISODate("2000-10-00T00:00:00Z"),
        site: "site-1",
        page: "/apache_pb.gif" },
    daily: {
        "1": 5445326,
        "2": 5214121,
        ... }
}
```

To read more about Pre-Aggregated data with MongoDB, please refer to mongoDB documentation:

- http://docs.mongodb.org/manual/use-cases/pre-aggregated-reports/
- http://docs.mongodb.org/manual/use-cases/hierarchical-aggregation/

6.6 Preaggregate Design Pattern with Call Data

We explained previously why preaggregating is a huge performance gain for analytic reporting and how it reduces disk seeks on your aggregate queries, we will now show how we apply this pattern to our call data.

Our data are the CDR (Call Detail Records) which are pre-processed for type validation, after this sanitisation of the call data, we proceed to the pre=aggragation step. For this we create a new daily_cdr collection which is aggregated daily.

Our code with PyMongo:

```
DAILY_ANALYTIC.update(
        {
        "_id": id_daily,
        "metadata": {
            "date": d,
            "switch_id": switch_id,
            "country_id": country_id,
            "accountcode": accountcode,
            "hangup_cause_id": hangup_cause_id,
            },
        },
        {
        "$inc": {
            "call_daily": 1,
            "call_hourly.%d" % (hour,): 1,
            "call_minute.%d.%d" % (hour, minute,): 1,
            "duration_daily": duration,
            "duration_hourly.%d" % (hour,): duration,
            "duration_minute.%d.%d" % (hour, minute,): duration,
            }
    }, upsert=True)
```

The '_id' is created with concatenation of the day, switch, country, accountcode and hangup cause ID.

The above collection is very fast to query, to retrieve the amount of calls for a day for a specific accountcode will be immediate. The field call_hourly can be used to plot the calls per hour for a single user or for a specific country.

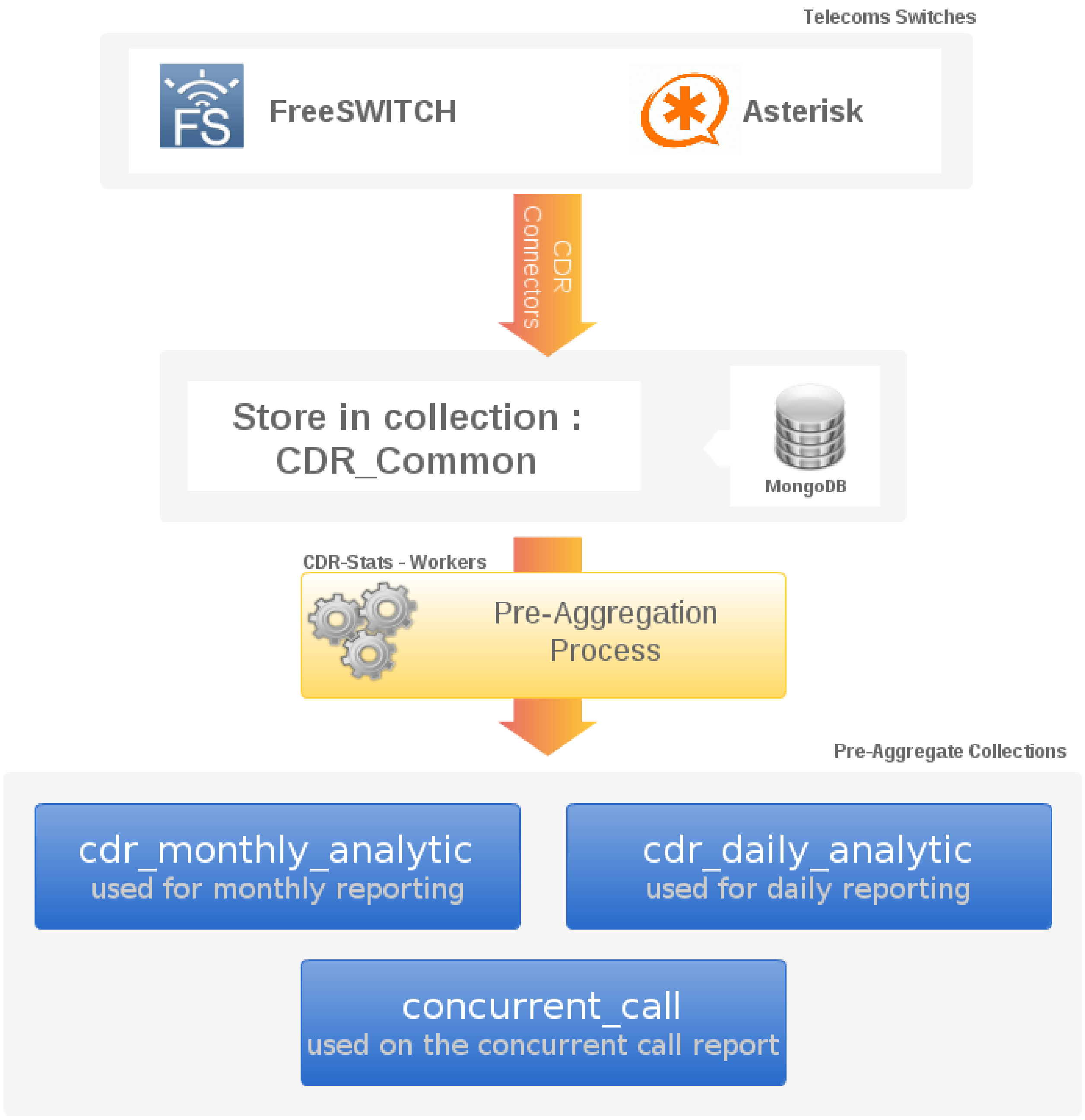
6.7 CDR-Stats MongoDB Collections

1) cdr_common: To collect all CDR's from different switches & store into one common format which include the following fields switch_id, caller_id_number, caller_id_name, destination_number, duration, billsec, hangup_cause_id, accountcode, direction, uuid, remote_media_ip, start_uepoch, answer_uepoch, end_uepoch, mduration, billmsec, read_codec, write_codec, cdr_type, cdr_object_id, country_id, authorized. This cdr_common collection used to view cdr records on customer panel

2) monthly_analytic: To collect monthly analytics from CDR's which include following fields date, country_id, accountcode, switch_id, calls, duration. This monthly_analytic collection is used to view monthly graph on customer panel

3) daily_analytic: To collect daily analytics from CDR's which include following fields, date, hangup_cause_id, country_id, accountcode, switch_id, calls, duration. This daily_analytic collection used to view daily graph/hourly graph on customer panel.

4) concurrent_call: To collect concurrent calls which include following fields switch_id, call_date, numbercall, accountcode. This concurrent_call collection is used to view concurrent call real-time graph on customer panel


Telecoms Switches
FreeSWITCH
Asterisk
CDR Connectors
Store in collection : CDR_Common
MongoDB
CDR-Stats - Workers
Pre-Aggregation Process
Pre-Aggregate Collections
cdr_monthly_analytic
used for monthly reporting
cdr_daily_analytic
used for daily reporting
concurrent_call
used on the concurrent call report

CHAPTER

SEVEN

DEVELOPER DOC

Contents:

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/
- MongoDB : http://www.mongodb.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                  - The code for APIs
|   `-- api_playground
|-- cdr                  - The code for CDR
|   |-- management
|   |-- templatetags
|   `-- fixtures
|-- cdr_alert            - The code for alarm, blacklist, whitelist
|   |-- management
|   `-- fixtures
|-- frontend             - The code for login, logout user
|-- user_profile         - The code for user detail of system
|-- 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`

class `cdr.models.`**`Switch`**`(`**args*, ***kwargs*`)`

This defines the Switch

Attributes:

- `name` - Name of switch.
- `ipaddress` - ipaddress

Name of DB table: voip_switch

7.3.2 `HangupCause`

class `cdr.models.`**`HangupCause`**`(`**args*, ***kwargs*`)`

This defines the HangupCause

Attributes:

- `code` - ITU-T Q.850 Code.
- `enumeration` - Enumeration
- `cause` - cause
- `description` - cause description

Name of DB table: hangup_cause

7.3.3 `UserProfile`

class `user_profile.models.`**`UserProfile`**`(`**args*, ***kwargs*`)`

This defines extra features for the user

Attributes:

- `accountcode` - Account name.
- `address` -
- `city` -
- `state` -

- `address` -
- `country` -
- `zip_code` -
- `phone_no` -
- `fax` -
- `company_name` -
- `company_website` -
- `language` -
- `note` -

Relationships:

- `user` - Foreign key relationship to the User model.
- `userprofile_gateway` - ManyToMany
- `userprofile_voipservergroup` - ManyToMany
- `dialersetting` - Foreign key relationship to the DialerSetting model.

Name of DB table: user_profile

7.3.4 `Alarm`

class `cdr_alert.models.`**`Alarm`**(**args*, ***kwargs*)

This defines the Alarm

Attributes:

- `user` -
- `name` - Alarm name
- `period` - Day | Week | Month
- `type` - ALOC (average length of call) ; ASR (answer seize ratio)
- `alert_condition` -
- `alert_value` - Input the value for the alert
- `alert_condition_add_on` -
- `status` - Inactive | Active
- `email_to_send_alarm` - email_to

Name of DB table: alert

7.3.5 `AlertRemovePrefix`

class `cdr_alert.models.`**`AlertRemovePrefix`**(**args*, ***kwargs*)

This defines the Alert Remove Prefix Define the list of prefixes that need to be removed from the dialed digits, assuming the phone numbers are in the format 5559004432, with the signifcant digits 9004432, the prefix 555 needs to be removed to analyse the phone numbers.

Attributes:

- `label` - Label for the custom prefix
- `prefix` - Prefix value

Name of DB table: alarm

7.3.6 `AlarmReport`

class `cdr_alert.models.`**`AlarmReport`**(**args*, ***kwargs*)

This defines the Alarm report

Attributes:

- `alarm` - Alarm name
- `calculatedvalue` - Input the value for the alert
- `daterun` -

Name of DB table: alert_report

7.3.7 `Blacklist`

class `cdr_alert.models.`**`Blacklist`**(**args*, ***kwargs*)

This defines the Blacklist

Attributes:

- `user` -
- `phonenumber_prefix` -
- `country` -

Name of DB table: alert_blacklist

7.3.8 `Whitelist`

class `cdr_alert.models.`**`Whitelist`**(**args*, ***kwargs*)

This defines the Blacklist

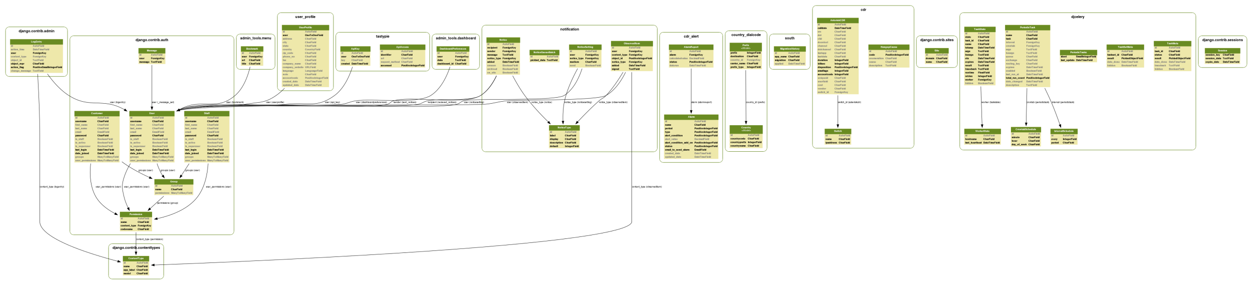
Attributes:

- `user` -
- `phonenumber_prefix` -
- `country` -

Name of DB table: alert_whitelist

7.4 Database Design

The current database schema is shown below:



Follow this link for more details : https://github.com/Star2Billing/cdr-stats/raw/master/docs/source/_static/images/model_cdr-stats.png

7.5 CDR-Stats Views

7.5.1 `index`

`cdr.views.`**`index`**`(`*request*`)`

Index Page of CDR-Stats

Attributes:

- `template` - frontend/index.html
- `form` - loginForm

7.5.2 `cdr_view`

`cdr.views.`**`cdr_view`**`(`*request*, **args*, ***kwargs*`)`

Caller.

7.5.3 `cdr_detail`

`cdr.views.`**`cdr_detail`**`(`*request*, **args*, ***kwargs*`)`

Detail of Call

Attributes:

- `template` - frontend/cdr_detail.html

Logic Description:

get the single call record in detail from mongodb collection

7.5.4 `cdr_dashboard`

`cdr.views.`**`cdr_dashboard`**`(`*request*, **args*, ***kwargs*`)`

Caller.

7.5.5 `cdr_overview`

`cdr.views.`**`cdr_overview`**`(`*request*, **args*, ***kwargs*`)`

Caller.

7.5.6 `cdr_realtime`

`cdr.views.`**`cdr_realtime`**(*request*, **args*, ***kwargs*)
Caller.

7.5.7 `cdr_report_by_hour`

`cdr.views.`**`cdr_report_by_hour`**(*request*, **args*, ***kwargs*)
Caller.

7.5.8 `cdr_concurrent_calls`

`cdr.views.`**`cdr_concurrent_calls`**(*request*, **args*, ***kwargs*)
Caller.

7.5.9 `world_map_view`

`cdr.views.`**`world_map_view`**(*request*, **args*, ***kwargs*)
Caller.

7.5.10 `customer_detail_change`

`user_profile.views.`**`customer_detail_change`**(*request*, **args*, ***kwargs*)
User Detail change on Customer UI

Attributes:

- `form` - UserChangeDetailForm, UserChangeDetailExtendForm, PasswordChangeForm
- `template` - 'frontend/registration/user_detail_change.html'

Logic Description:

- User is able to change their details.

7.5.11 `alarm_list`

`cdr_alert.views.`**`alarm_list`**(*request*, **args*, ***kwargs*)
Alarm list for the logged in user

Attributes:

- `template` - frontend/cdr_alert/alert_list.html

Logic Description:

- List all alarms which belong to the logged in user.

7.5.12 `alarm_add`

`cdr_alert.views.`**`alarm_add`**(*request*, **args*, ***kwargs*)
Add new Alarm for the logged in user

Attributes:

- `form` - AlarmForm
- `template` - frontend/cdr_alert/alarm/change.html

Logic Description:

- Add a new Alarm which will belong to the logged in user via the AlarmForm & get redirected to the Alarm list

7.5.13 `alarm_del`

`cdr_alert.views.`**`alarm_del`**(*request*, **args*, ***kwargs*)
Delete a alarm for a logged in user

Attributes:

- `object_id` - Selected alarm object
- `object_list` - Selected alarm objects

Logic Description:

- Delete selected the alarm from the alarm list

7.5.14 `alarm_change`

`cdr_alert.views.`**`alarm_change`**(*request*, **args*, ***kwargs*)
Update/Delete Alarm for the logged in user

Attributes:

- `object_id` - Selected alarm object
- `form` - AlarmForm
- `template` - frontend/cdr_alert/alarm/change.html

Logic Description:

- Update/delete selected alarm from the alarm list via alarmForm & get redirected to alarm list

7.5.15 `alert_report`

`cdr_alert.views.`**`alert_report`**(*request*, **args*, ***kwargs*)
To get alarm report for logged in user

Attributes:

- `form` - AlarmReportForm
- `template` - frontend/cdr_alert/alarm_report.html

7.6 CDR-Stats Tasks

7.6.1 `sync_cdr_pending`

class `cdr.tasks.`**`sync_cdr_pending`**

A periodic task that checks for pending CDR to import

`run` (**args*, ***kwargs*)

Caller.

7.6.2 `chk_alarm`

class `cdr_alert.tasks.`**`chk_alarm`**

A periodic task to determine unusual call patterns.

Sends an email if an alert condition is matched.

Usage:

chk_alarm.delay()

7.6.3 `blacklist_whitelist_notification`

class `cdr_alert.tasks.`**`blacklist_whitelist_notification`**

Send email notification whne destination number matched with blacklist or whitelist.

Usage:

blacklist_whitelist_notification.delay(notice_type)

7.6.4 `send_cdr_report`

class `cdr_alert.tasks.`**`send_cdr_report`**

A periodic task to send previous day's CDR Report as mail

Usage:

send_cdr_report.delay()

`run` (**args*, ***kwargs*)

Caller.

7.7 Test Case Descriptions

7.7.1 Requirement

Run/Start Celery:

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

or:

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

Run/Start Redis:

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

7.7.2 How to Run Tests

1. Run Full Test Suit:

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

2. Run CDRStatsTastypieApiTestCase:

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

3. Run CDRStatsAdminInterfaceTestCase:

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

4. Run CDRStatsCustomerInterfaceTestCase:

```
$ 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.
- Flot is a pure JavaScript plotting library for jQuery, with a focus on simple usage, attractive looks and interactive features.
- 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

EIGHT

API REFERENCE

Contents:

8.1 SwitchResource

class api.switch_api.**SwitchResource**(*api_name=None*)

Attributes Details:

- name - Name of switch.
- ipaddress - ipaddress

Create:

CURL Usage:

```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST --data
```

Response:

```
HTTP/1.0 201 CREATED
Date: Fri, 23 Sep 2011 06:08:34 GMT
Server: WSGIServer/0.1 Python/2.7.1+
Vary: Accept-Language, Cookie
Content-Type: text/html; charset=utf-8
Location: http://localhost:8000/api/app/switch/1/
Content-Language: en-us
```

Read:

CURL Usage:

```
curl -u username:password -H 'Accept: application/json' -X GET http://localhost:8000/api/v1/
```

8.2 CdrDailyResource

class api.cdr_daily_api.**CdrDailyResource**(*api_name=None*)

Attributes Details:

- _id - contact id
- start_uepoch - call date

- `destination_number` - destination
- `hangup_cause_id` -
- `switch_id` - switch

Read:

CURL Usage:

```
curl -u username:password -H 'Accept: application/json' -X POST --data '{"start_uepoch":"201
```

Response:

```
[
   {
      "_id":"4f3dec808365701c4a25aaad",
      "accountcode":"1000",
      "destination_number":"5545",
      "hangup_cause_id":8,
      "start_uepoch":"2012-02-15T00:00:00",
      "switch_id":1
   },
   {
      "_id":"4f3dec808365701c4a25aab0",
      "accountcode":"1000",
      "destination_number":"2133",
      "hangup_cause_id":9,
      "start_uepoch":"2012-02-15T00:00:00",
      "switch_id":1
   }
]
```

8.3 CdrResource

class `api.cdr_api.`**`CdrResource`**(*api_name=None*)

API to bulk create cdr

Attributes:

- `accountcode` -
- `answer_uepoch` -
- `billmsec` -
- `billsec` -
- `caller_id_name` -
- `caller_id_number` -
- `cdr_object_id` -
- `cdr_type` -
- `destination_number` -
- `direction":"inbound` -
- `duration` -
- `end_uepoch` -

- `hangup_cause_id` -
- `mduration` -
- `read_codec` -
- `remote_media_ip` -
- `start_uepoch` -
- `switch_id` -
- `uuid`
- `write_codec` -

Validation:

- CdrValidation()

CURL Usage:

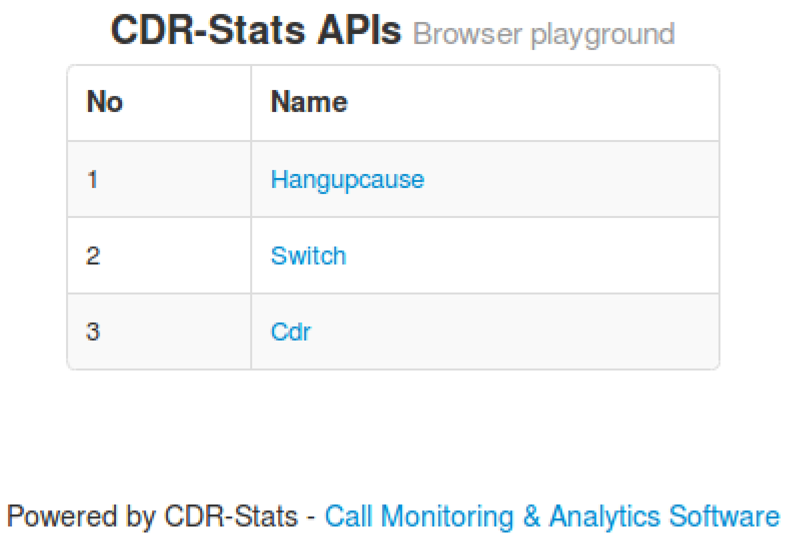
```
curl -u username:password --dump-header - -H "Content-Type:application/json" -X POST --data '{"s
```

Response:

```
{
    "_id":"4f3dec801d41c80b8e000000",
    "accountcode":"1000",
    "answer_uepoch":"2012-01-25T14:05:53",
    "billmsec":"12960",
    "billsec":13,
    "caller_id_name":"1000",
    "caller_id_number":"1000",
    "cdr_object_id":"4f3dec231d41c80b2600001f",
    "cdr_type":1,
    "destination_number":"5545",
    "direction":"inbound",
    "duration":107,
    "end_uepoch":"2012-01-25T14:06:06",
    "hangup_cause_id":8,
    "mduration":"12960",
    "read_codec":"G722",
    "remote_media_ip":"192.168.1.21",
    "start_uepoch":"2012-02-15T22:02:51",
    "switch_id":1,
    "uuid":"2ffd8364-592c-11e1-964f-000c296bd875",
    "write_codec":"G722"
}
```

Testing console of APIs:

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



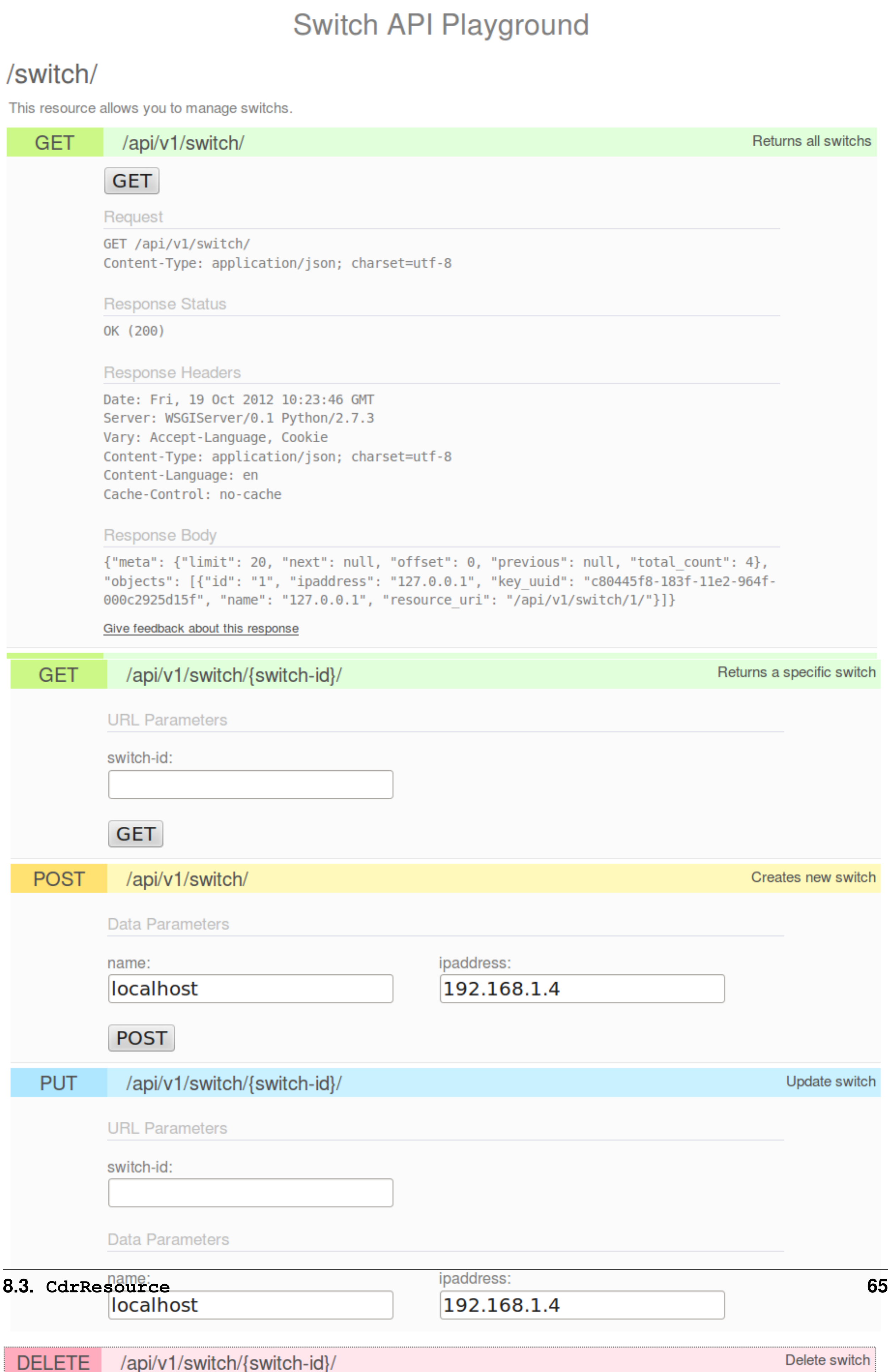
CDR-Stats APIs Browser playground

No	Name
1	Hangupcause
2	Switch
3	Cdr

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 switchs.

GET /api/v1/switch/ Returns all switchs

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": "c80445f8-183f-11e2-964f-
000c2925d15f", "name": "127.0.0.1", "resource_uri": "/api/v1/switch/1/"}]}
```

Give feedback about this response

GET /api/v1/switch/{switch-id}/ Returns a specific switch

URL Parameters

switch-id:

POST /api/v1/switch/ Creates new switch

Data Parameters

name: localhost

ipaddress: 192.168.1.4

PUT /api/v1/switch/{switch-id}/ Update switch

URL Parameters

switch-id:

Data Parameters

name: localhost

ipaddress: 192.168.1.4

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

CHAPTER NINE

CONTRIBUTING

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.

The Code of Conduct is heavily based on the Ubuntu Code of Conduct, Celery Code of Conduct, and the Pylons Code of Conduct.

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 many 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.

- CDR-Stats: http://github.com/Star2Billing/cdr-stats/issues/
- Celery: https://github.com/ask/celery/issues/
- Freeswitch: http://jira.freeswitch.org/secure/Dashboard.jspa
- Asterisk: http://issues.asterisk.org/jira/

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.

pep8.py is a utility you can use to verify that your code is following the conventions.

- Docstrings must follow the PEP-257 conventions, and use the following style.

  Do this:

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

      More details.

      """
  ```

  or:

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

  but not this:

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

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

CHAPTER

TEN

RESOURCES

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/Star2Billing/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/

Beginner's Guide : http://www.cdr-stats.org/documentation/beginners-guide/

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.

CHAPTER

ELEVEN

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 drop your current mongoDB, you can do this with this command:

```
$ mongo cdr-stats --eval 'db.dropDatabase();'
```

Update all your CDRs to be reimported as we flag them after import. This next step is dependant on your CDR store,

Mysql with Asterisk: run this command on the CDR Database:

```
$ UPDATE  cdr SET  import_cdr =  '0';
```

MongoDB with Freeswitch: Run this command in MongoDB

$ use freeswitch_cdr; db.cdr.update({“import_cdr” : 1}, { $set : {“import_cdr” : 0}}, { multi: true });

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:

```
$ cd /usr/share/cdr_stats
$ 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

TWELVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX