CDR-Stats Documentation

Release 2.0.alpha

Arezqui Belaid

May 10, 2012

CONTENTS

Version 2.0

Release 2.0.alpha

Date May 02, 2012

Contents:

–

CHAPTER

ONE

GETTING STARTED

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 you to browse and analyse CDR (Call Detail Records).

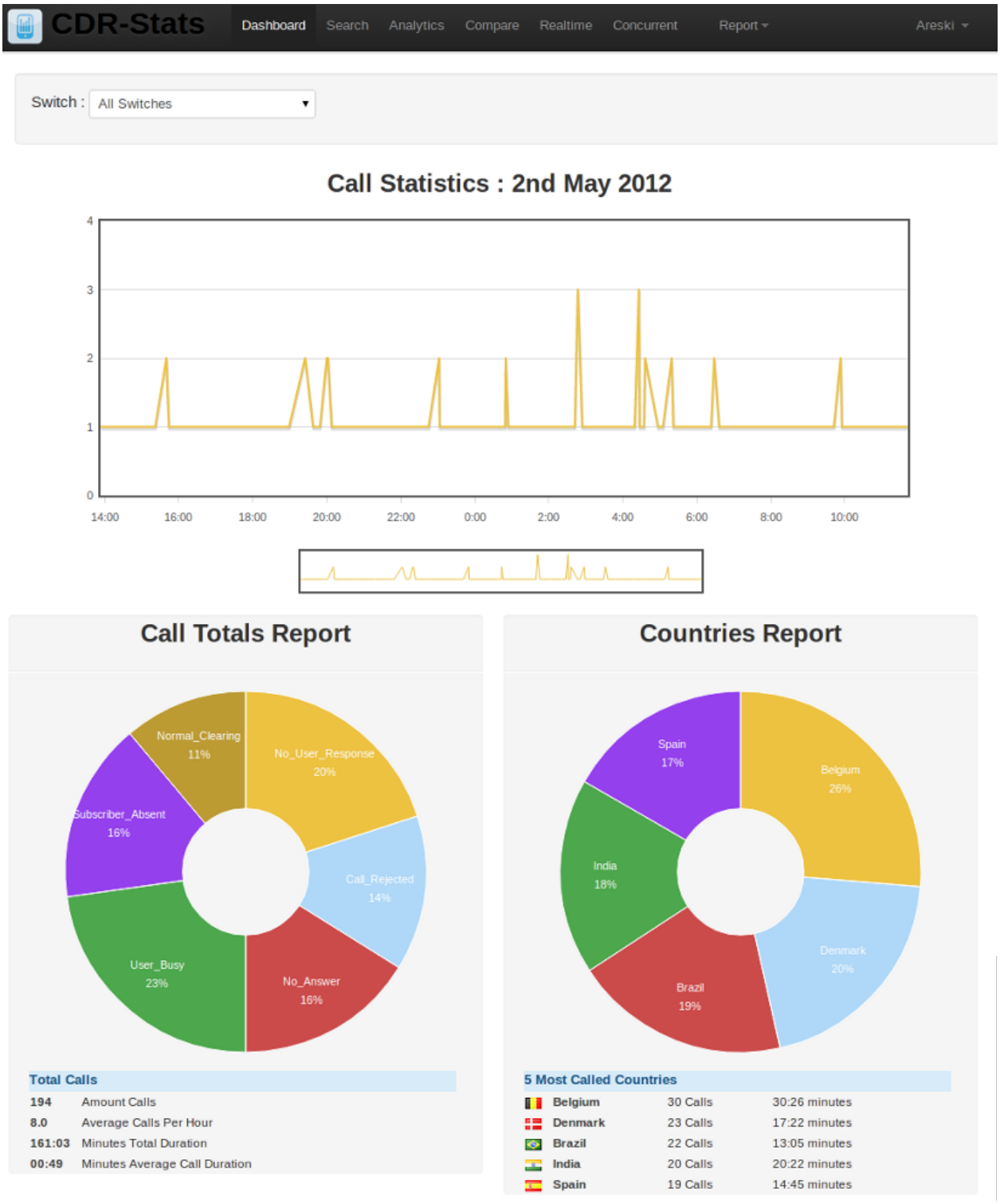
Different reporting tools are provided:

- Search CDR: Search, filter, display and export CDR.
- Monthly Report: Summarise and compare call traffic history month on month.
- Analyse CDR : Analyse and compare call volumes with the previous day's traffic.
- Daily Traffic : Graph and filter traffic loads by hour during the day.

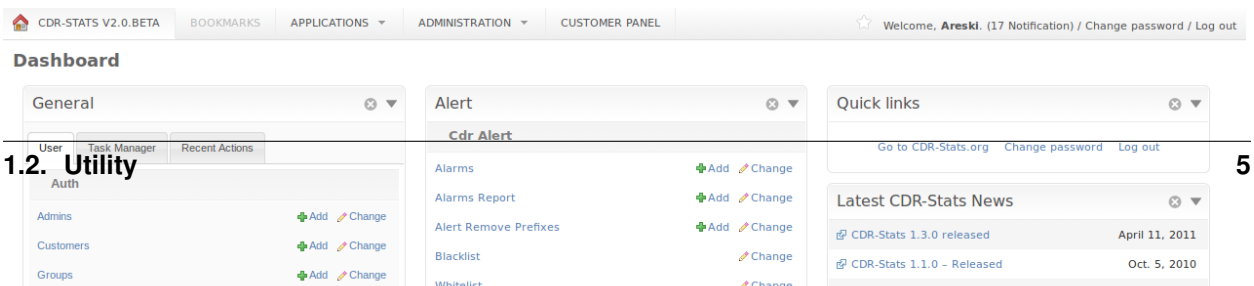
MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas. The goal of MongoDB is to bridge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

Version 2.0 of CDR-Stats supports Asterisk and Freeswitch 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.

1.1.1 Screenshot Dashboard



1.1.2 Screenshot Admin UI



through, can detect errors and failures, and alert the systems administrator is unexpected traffic is noted.

1.3 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.

One of the three popular databases (MySQL / Postgresql / SQLite) is used for for managing CDR-Stats, such as 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.

Freeswitch is supported using the mod_mongo module to write CDR directly into MongoDB. For other switches such as Asterisk, connectors can be built to connect to the switch's database store, such as MySQL, SQLite, or Postgresql.

Add graph on Architect

1.4 Features

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

- Visualise traffic which helps to identify unusual patterns.
- Map view, see where the traffic comes from and where it goes to
- Compare traffic to previous dates, see how your traffic evolves, and patterns change.
- Monitor VoIP server, set alerts to detect potential fraud
- Send daily mail reports of your VoIP traffic
- Traffic displayed in realtime
- Blacklist Phone number patterns to receive alarms
- Geographic alerts
- Multi-tenant, allowing many customer to monitor their own CDR on one instance of CDR-Stats
- Multi-switch, monitor traffic from many switches in one location

1.5 Latest documentation

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

CHAPTER

TWO

INSTALLATION

Contents:

2.1 Installation overview

2.1.1 Install requirements

A Requirements file gives you a way to create an environment where you can put all optional dependencies which are needed for the Project/Application.

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

- python >= 2.5 (programming language)
- Apache / http server with WSGI modules
- Django Framework >= 1.4 (Python based Web framework)
- Celery >= 2.2 (Asynchronous task queue/job queue based on distributed message passing)
- django-celery >= 2.2.4 (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 project
- 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

Use PIP to install all the requirements,:

```
$ pip install -r requirements.txt
```

2.1.2 Installation Script

You can install CDR-Stats manually or using the shell script provided.

To install CDR-Stats using the script,:

```
$ chmod +x install/install-cdrstats.sh
```

```
$ ./install/install-cdrstats.sh
```

```
$ chmod +x install/install-celery.sh
```

```
$ ./install/install-celery.sh
```

2.1.3 Running CDR-Stats

Inside CDR-Stats directory you should run:

```
$ mkdir database
```

```
$ python manage.py syncdb
```

```
$ python manage.py collectstatic
```

```
$ python manage.py runserver
```

`syncdb` will create a database named test.db in `database` folder of the CDR-Stats directory. We have configured CDR-Stats to do this, but you can change this simply by modifying settings.py where DATABASES dictionary is constructed. You can find 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.

`runserver` runs an embedded webserver to test your site. By default it will run on http://localhost:8008. This is configurable and more information can be found on `runserver` in Django documentation.

2.2 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.

2.2.1 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
```

2.2.2 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, you can continue reading Setting up RabbitMQ.

2.3 Celery Installation

2.3.1 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-0.0.0.tar.gz

$ cd celery-0.0.0

$ 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

USER GUIDE

Contents:

3.1 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 is used to monitor CDR for user defined unusual activity, and react by sending an alert email.

coming soon..

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 unu
- Graphical tools help detect unusual call patterns which may indicate suspicious or fraudulent activ
- 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.
```

3.2 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.

3.3 Admin Panel

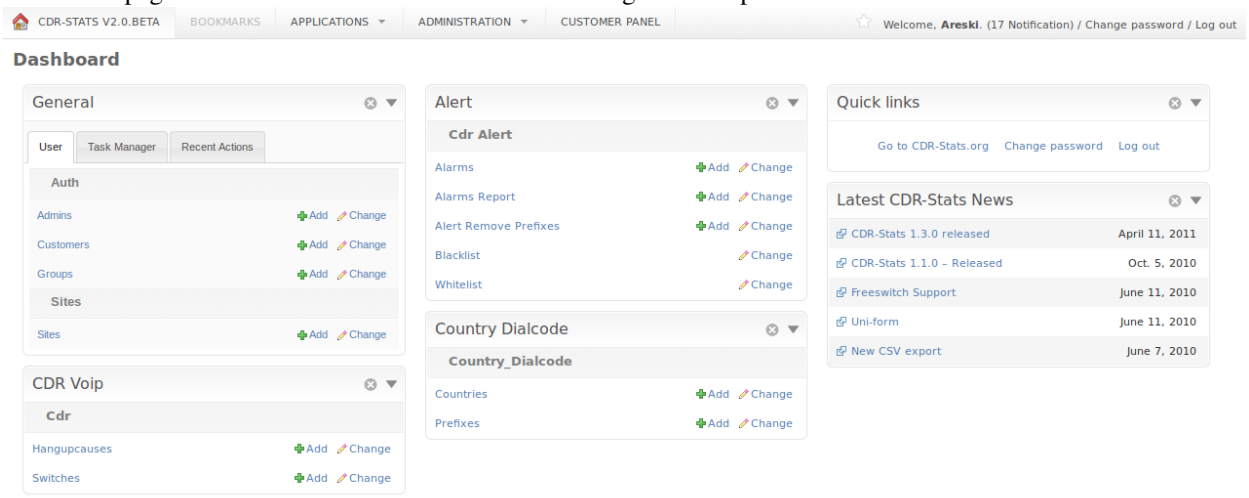
http://localhost:8008/admin/

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

3.3.1 Screenshot with Features

Dashboard

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

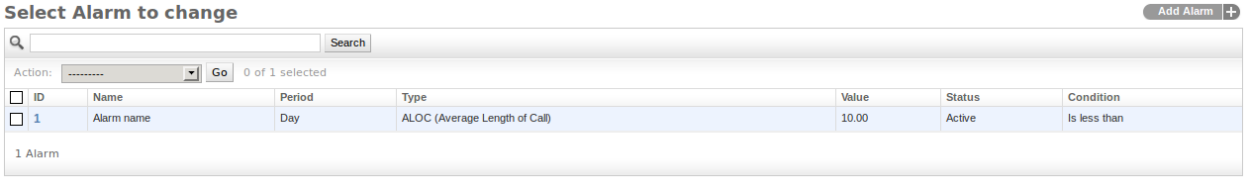


Alarm

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

URL:

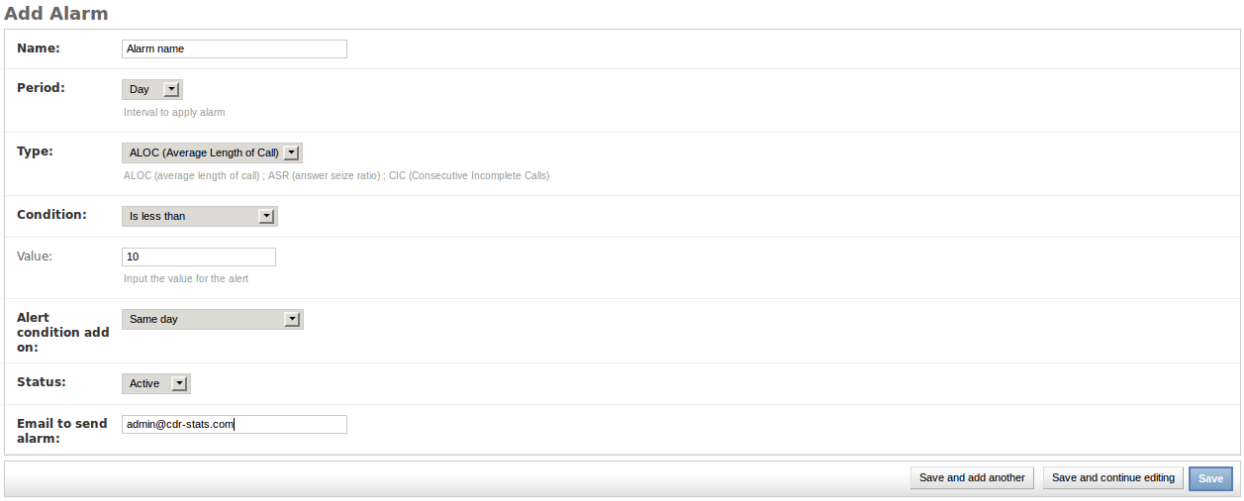
- http://localhost:8008/admin/cdr_alert/alarm/



To Add/Update alarm

URL:

- http://localhost:8008/admin/cdr_alert/alarm/add/
- http://localhost:8008/admin/cdr_alert/alarm/1/



Alarm-report

The alarmreport will be displayed from the following URL.

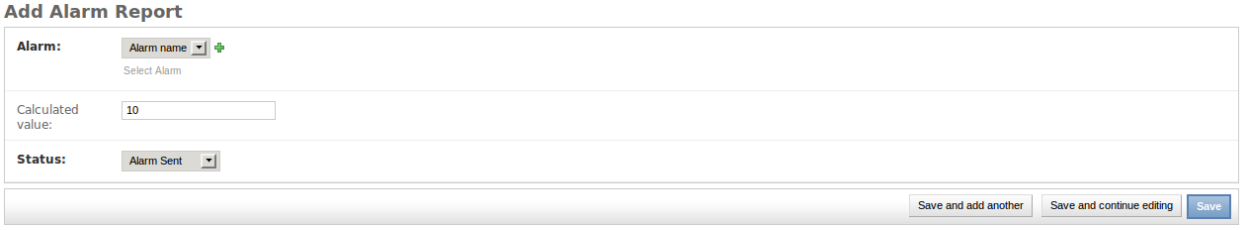
URL:

- http://localhost:8008/admin/cdr_alert/alarmreport/

To Add/Update alarmreport

URL:

- http://localhost:8008/admin/cdr_alert/alarmreport/add/
- http://localhost:8008/admin/cdr_alert/alarmreport/1/

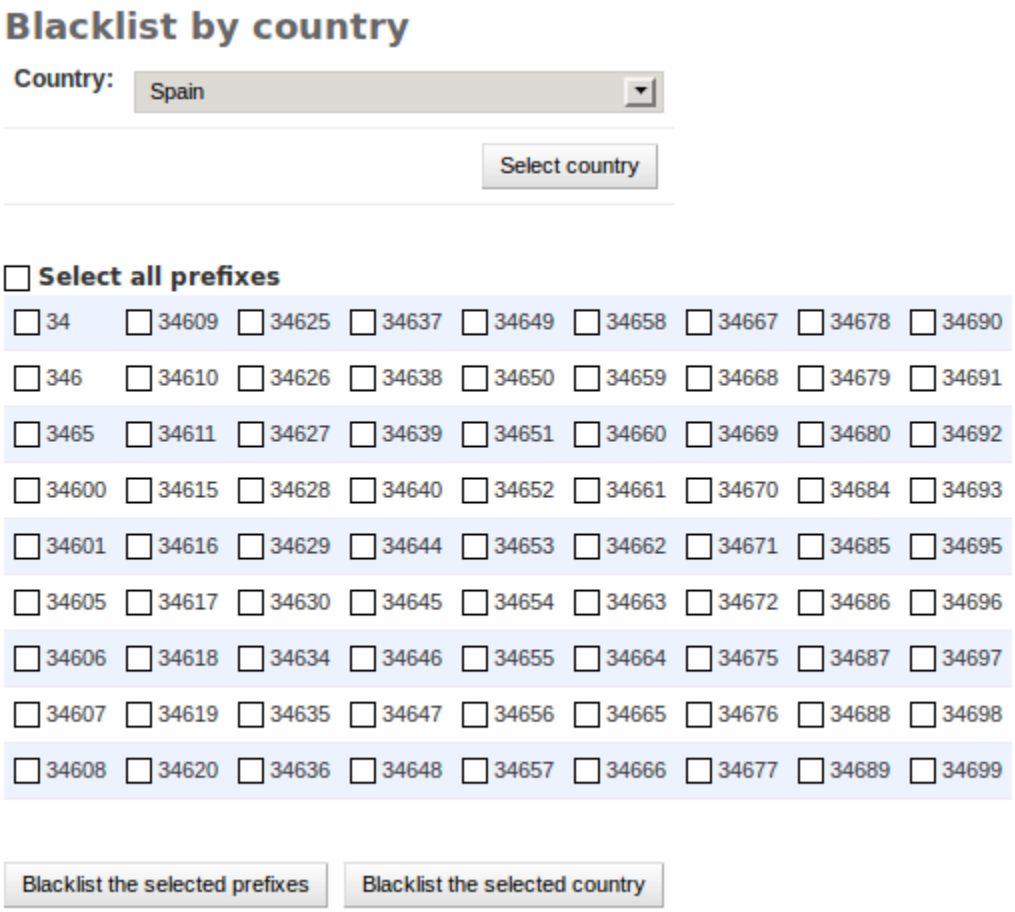


Blacklist

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

URL:

- http://localhost:8008/admin/cdr_alert/blacklist/



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:8008/admin/cdr_alert/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.

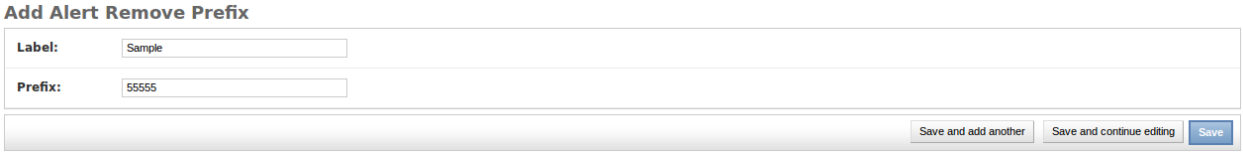
URL:

- http://localhost:8008/admin/cdr_alert/alertremoveprefix/

To Add/Update alert-removep-refix

URL:

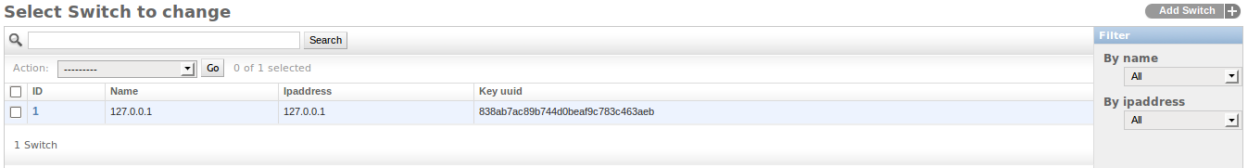
- http://localhost:8008/admin/cdr_alert/alertremoveprefix/add/
- http://localhost:8008/admin/cdr_alert/alertremoveprefix/1/



Switch

URL:

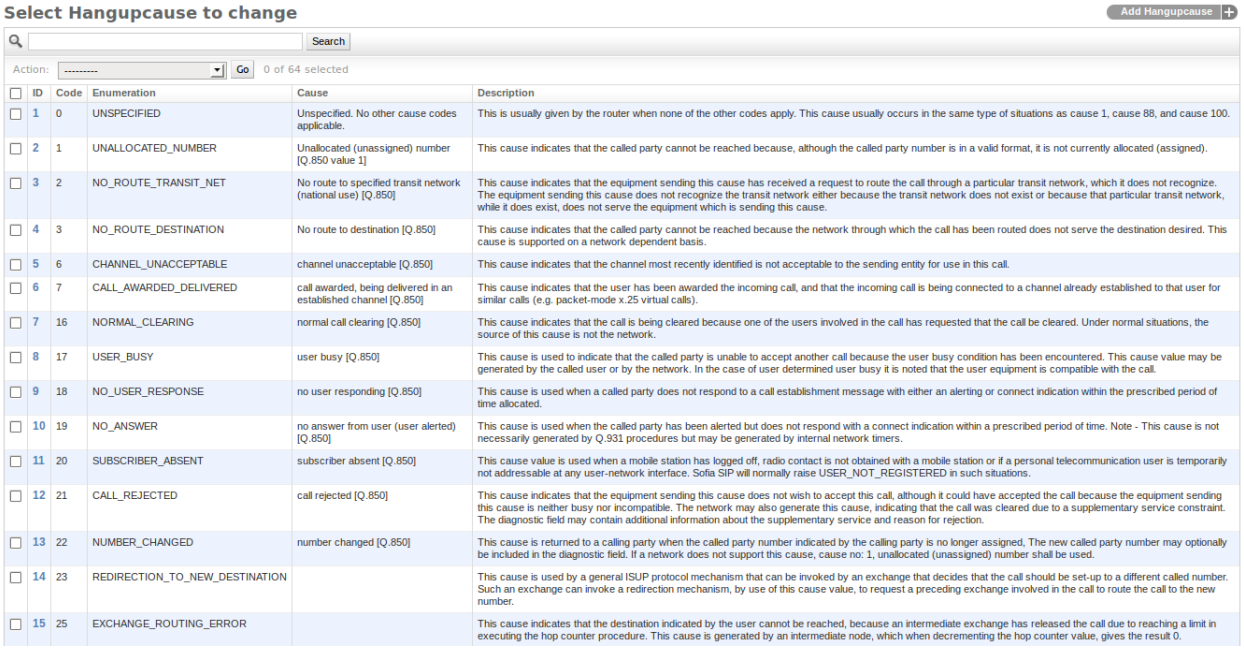
- http://localhost:8008/admin/cdr/switch/



HangupCause

URL:

- http://localhost:8008/admin/cdr/hangupcause/



Select Hangupcause to change

Add Hangupcause

Search

Action: --------- Go 0 of 64 selected

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.

3.4 Customer Panel

User Interface :

This application provides a user interface...

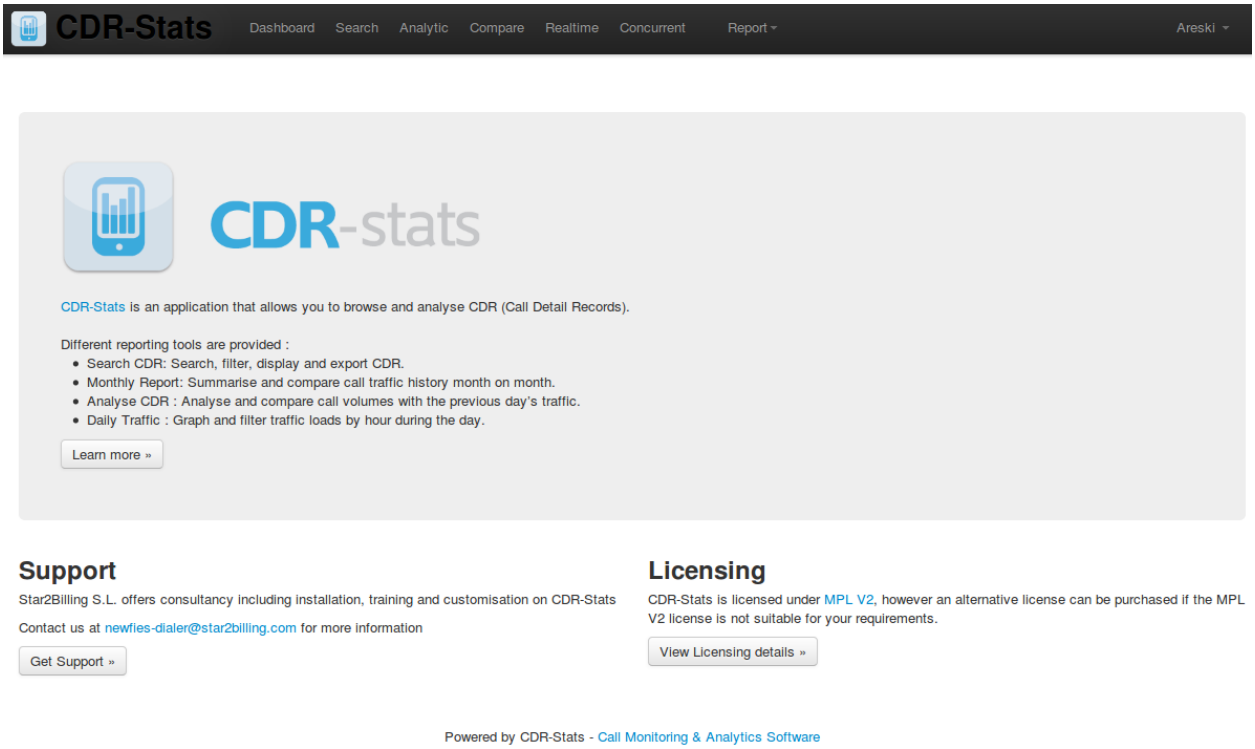
http://localhost:8008/

- Screenshot with Features

3.4.1 Screenshot with Features

Index

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

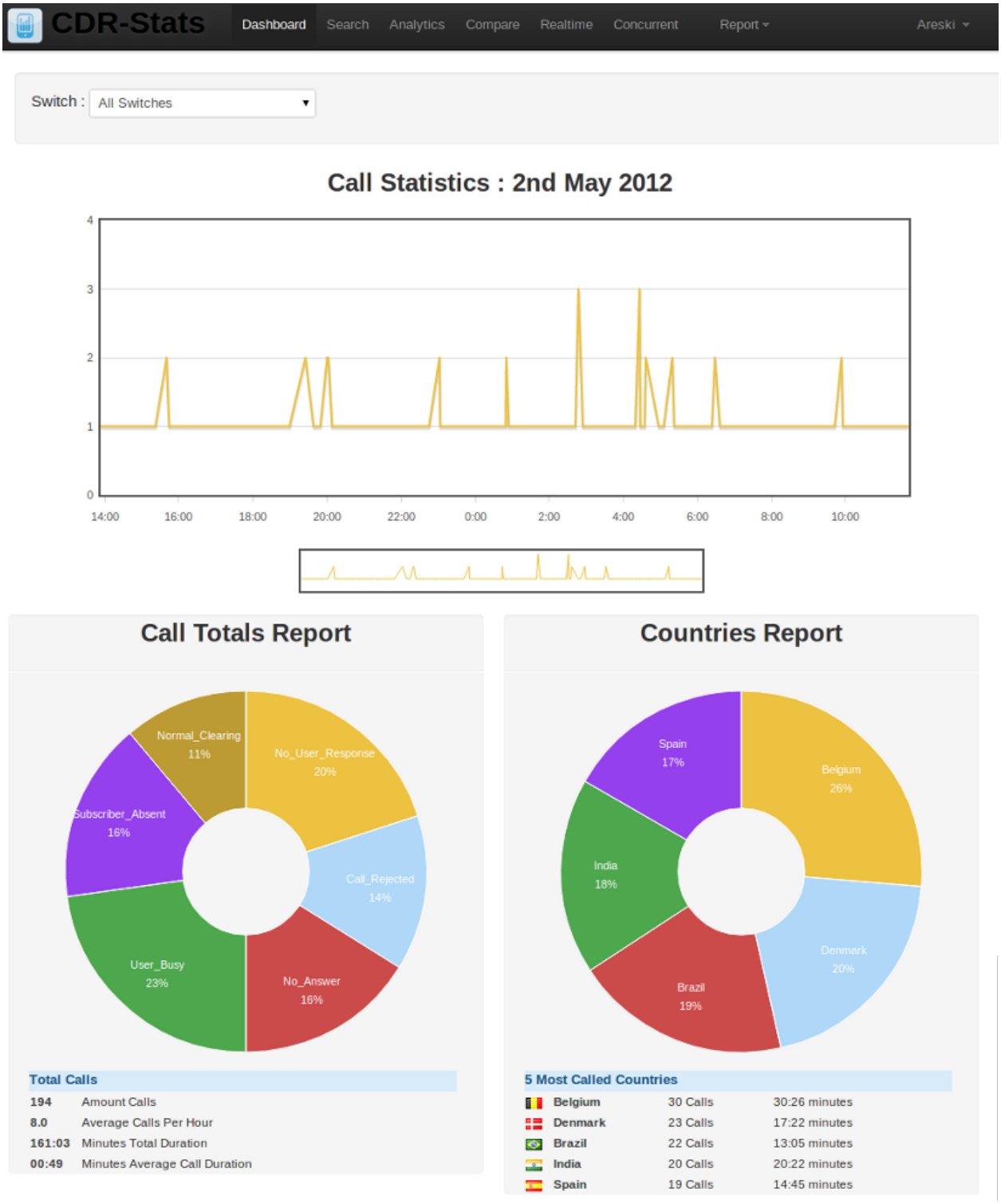


Dashboard

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

URL:

- http://localhost:8008/dashboard/



CDR-View

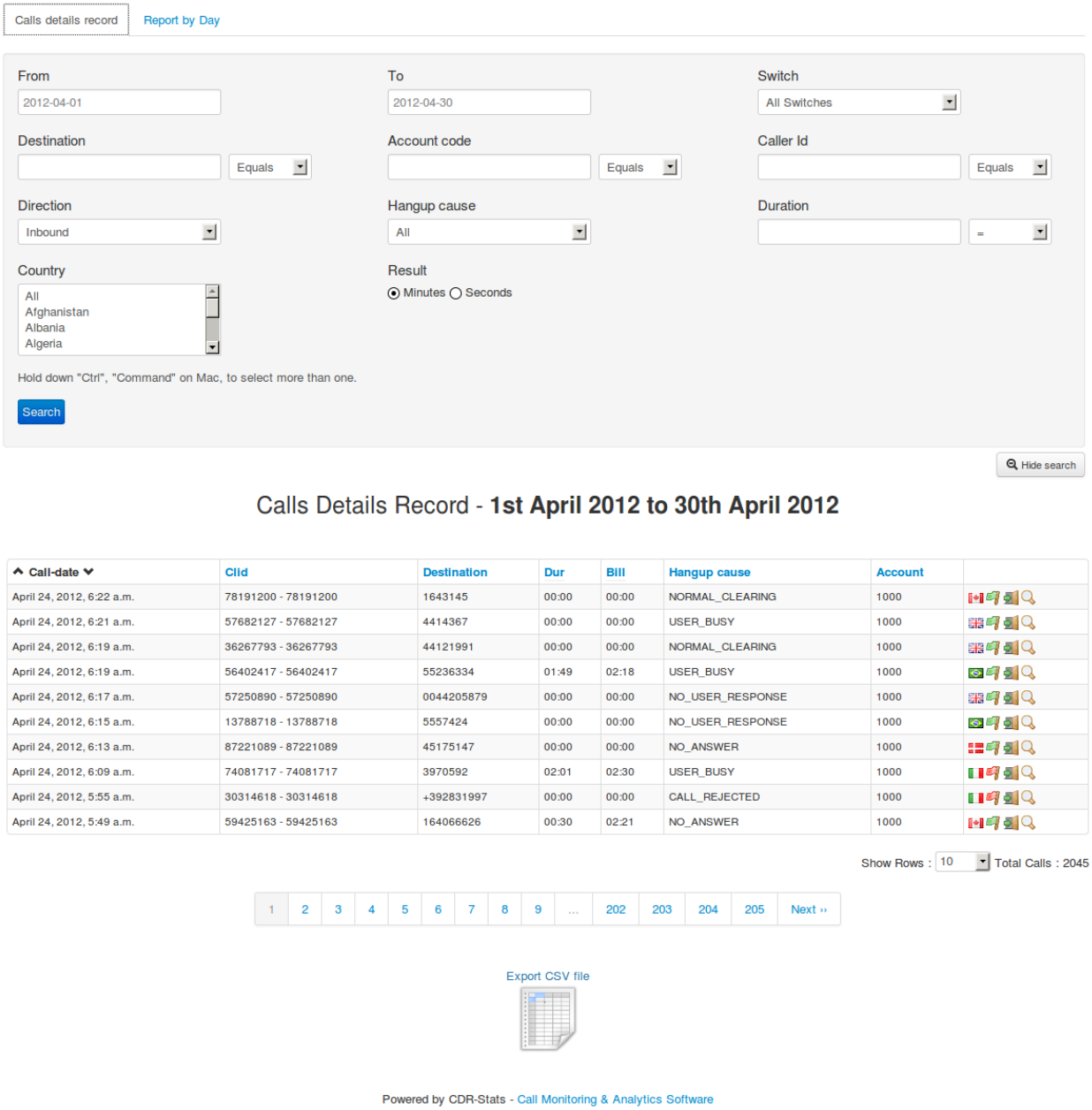
Call detail records listed in table format which can be exported to CSV file.

Advanced Search allows further filtering and searching on a range of criteria

The Report by Day shows a graphical illustration of the calls, minutes and average call time.

URL:

- http://localhost:8008/cdr_view/



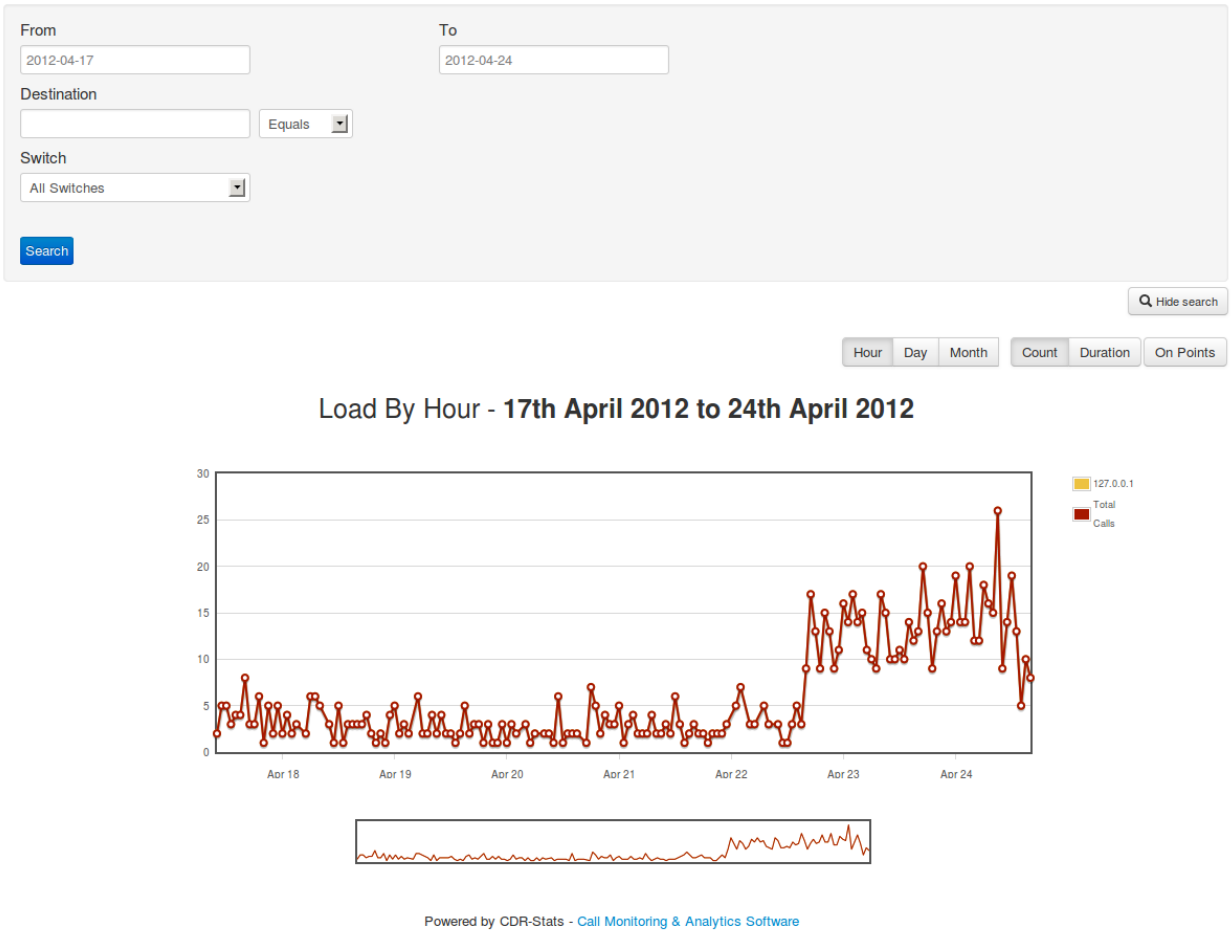
Date	Duration	Graphic	Calls	ACT
Tue 24 Apr 2012	140:40		170	00:49
Mon 23 Apr 2012	248:13		323	00:46
Sun 22 Apr 2012	165:43		194	00:51
Sat 21 Apr 2012	53:38		59	00:54
Fri 20 Apr 2012	39:38		64	00:37
Thu 19 Apr 2012	52:31		55	00:57
Wed 18 Apr 2012	47:50		67	00:42
Tue 17 Apr 2012	69:38		86	00:48
Mon 16 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

In this view, you can get pictorial view of calls with call-count or call-duration from any date or date-range

URL:

- http://localhost:8008/cdr_overview/

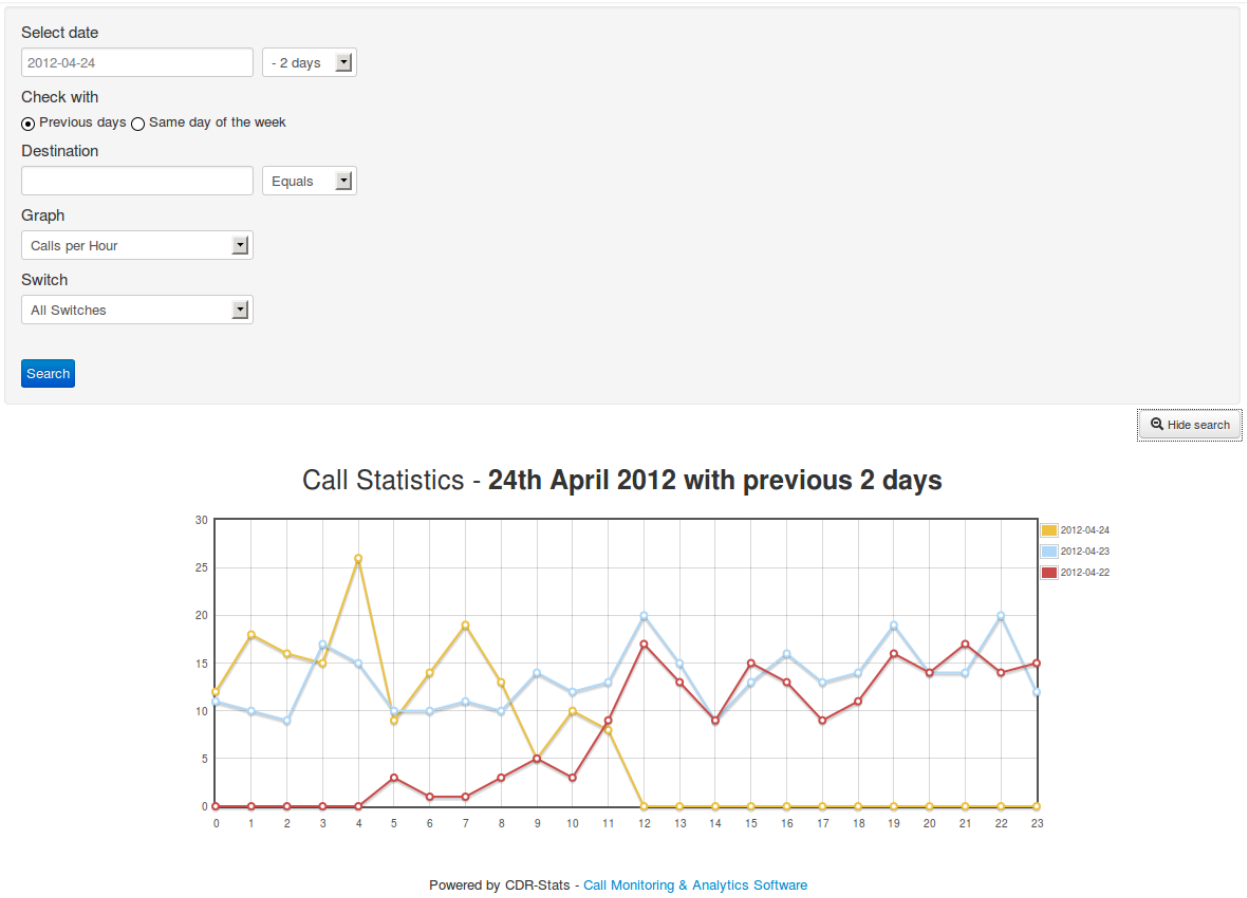


CDR-Hourly-Report

In this view, you can get hourly pictorial view of calls with call-count & call-duration. You can compare different dates

URL:

- http://localhost:8008/hourly_report/

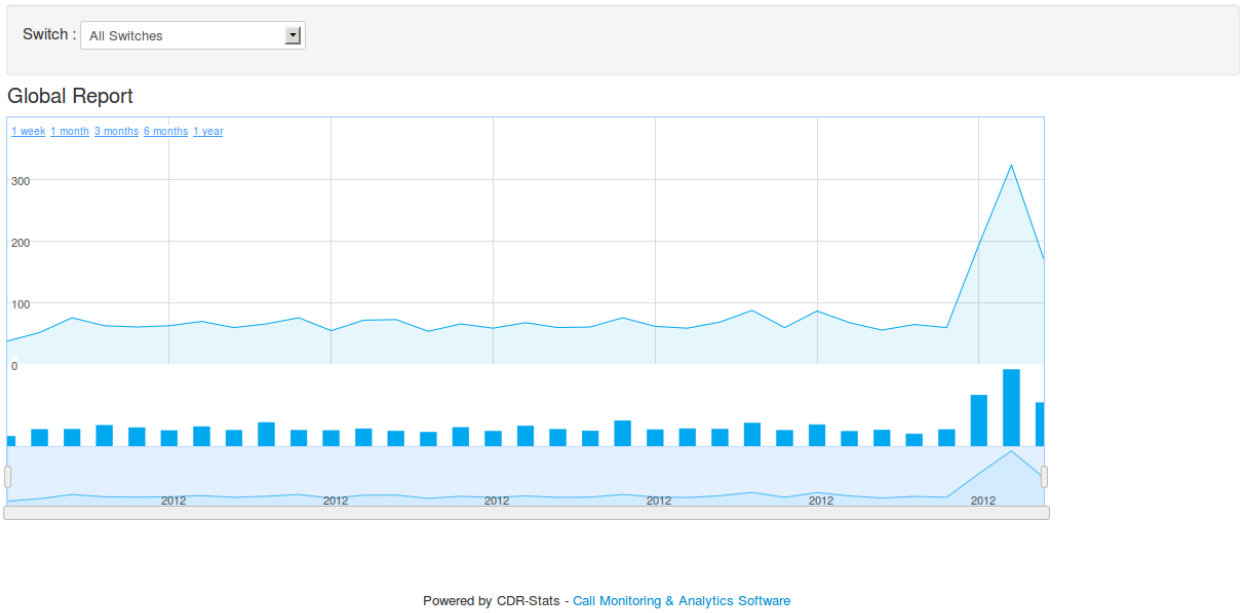


CDR-Global-Report

In this view, you can get pictorial view of all calls

URL:

- http://localhost:8000/global_report/

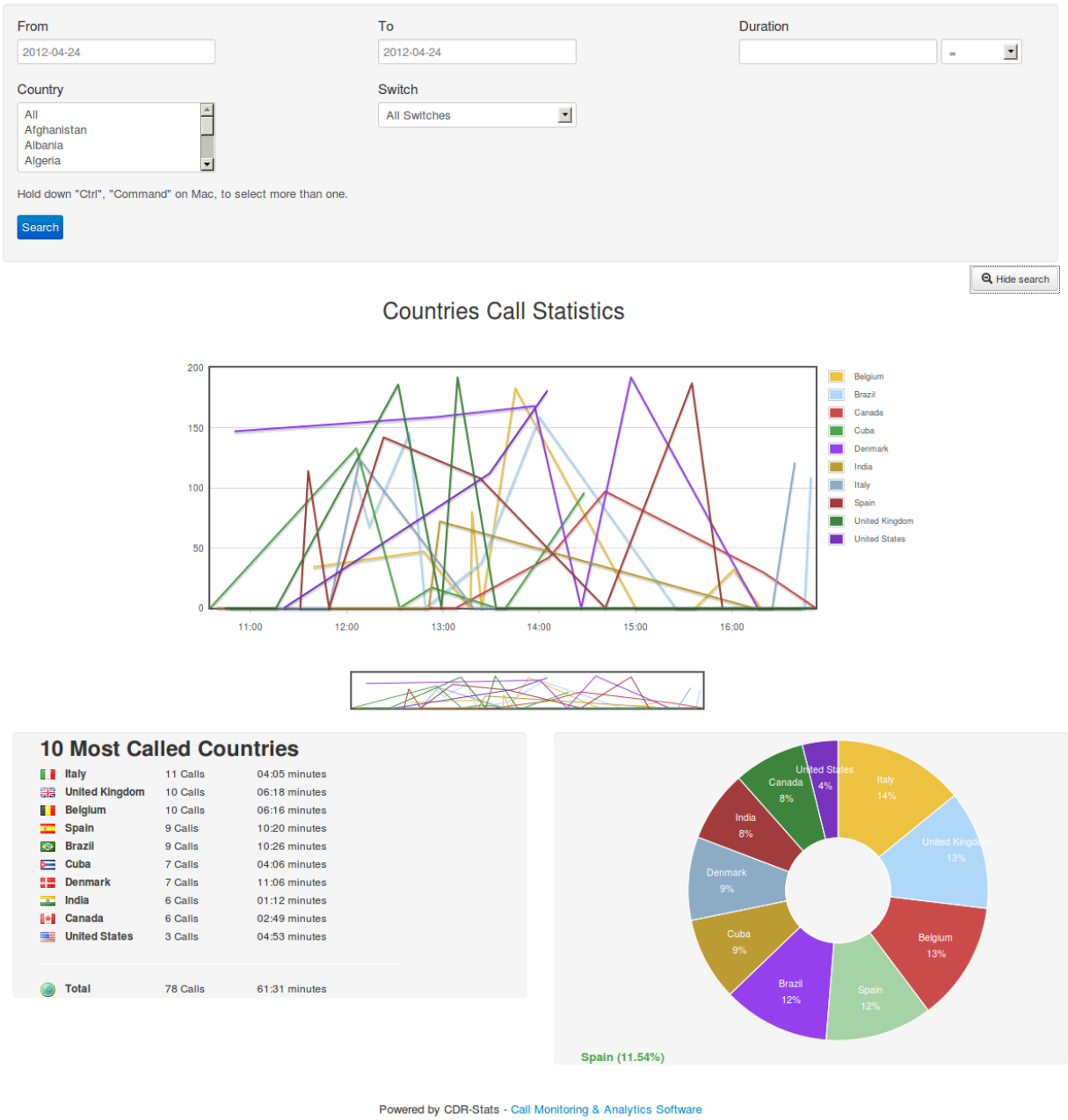


CDR-Country-Report

In this view, you can get pictorial view of all calls by country. Also you can have 10 most called countries name with pie chart

URL:

- http://localhost:8008/country_report/

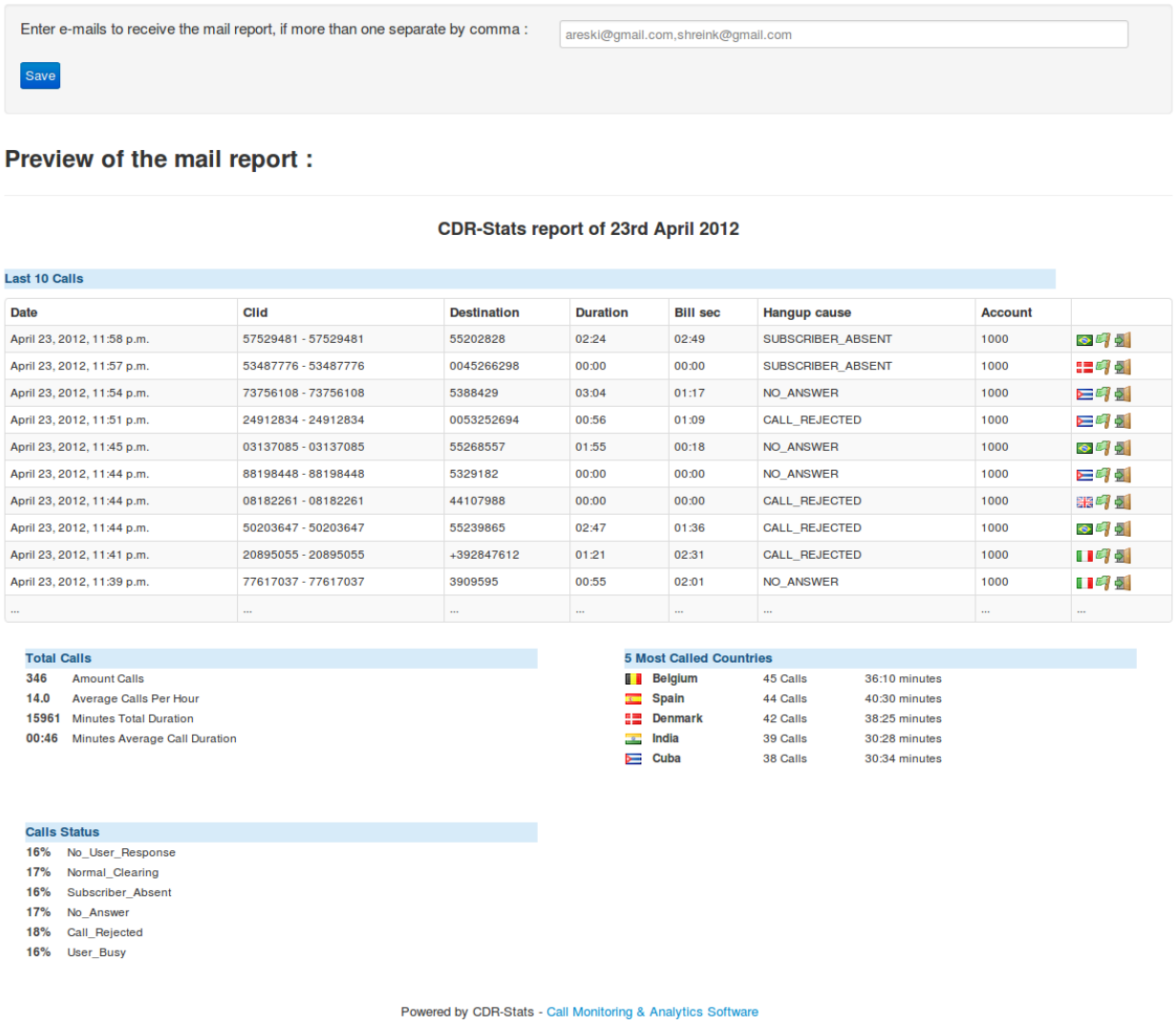


Mail-Report

In this view, there is 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.

URL:

- http://localhost:8008/mail_report/

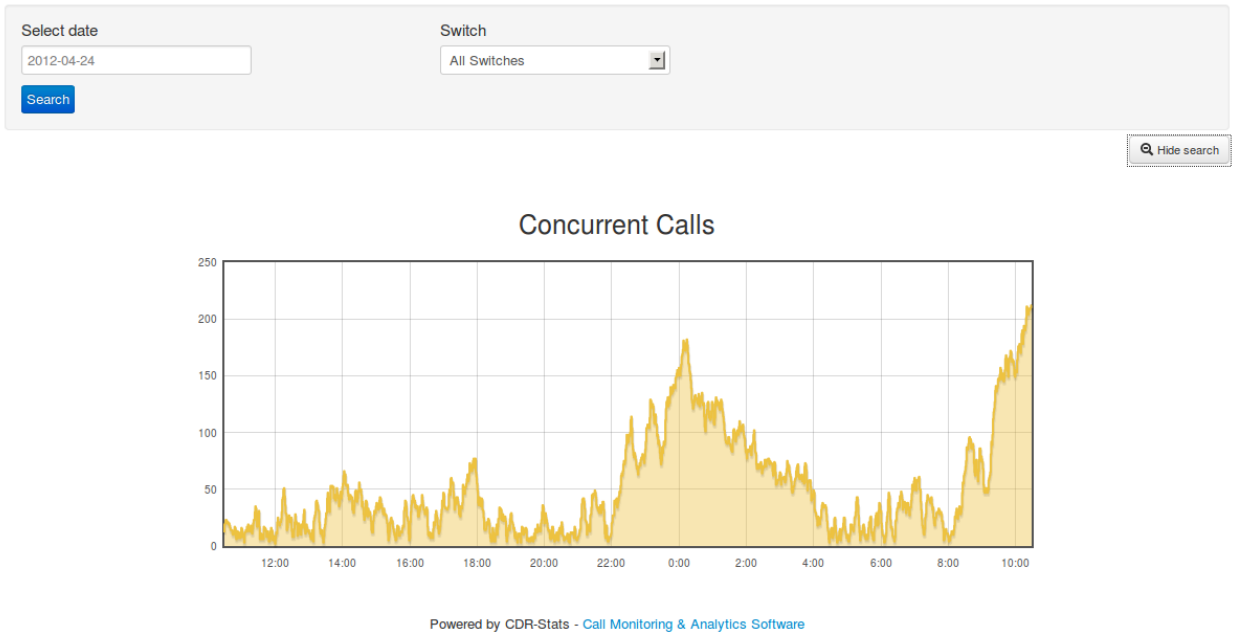


Concurrent-call-report

In this view, you can get report of concurrent calls

URL:

- http://localhost:8008/cdr_concurrent_calls/

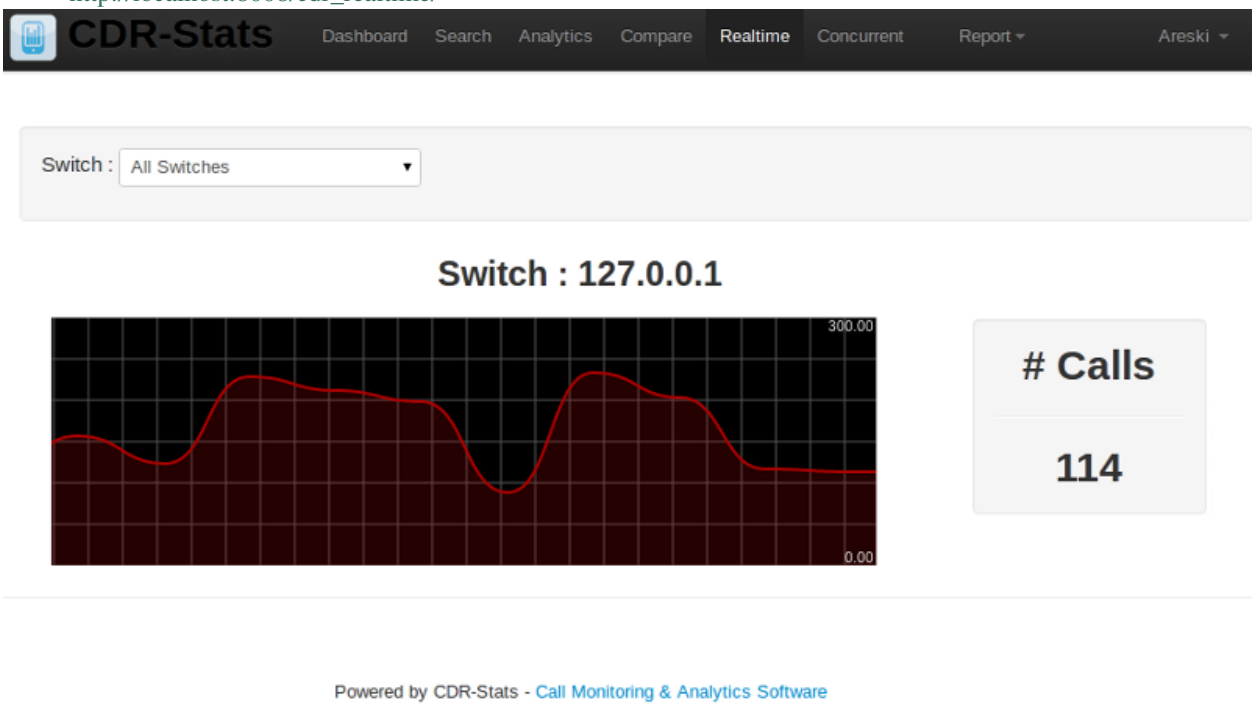


Realtime-Report

In this view, you can monitor in realtime the traffic on your voip servers. Currently, only Freeswitch is supported.

URL:

- http://localhost:8008/cdr_realtime/



CHAPTER

FOUR

CONFIGURATION AND DEFAULTS

Contents:

4.1 Sample Configuration

This is a sample configuration to get started should you chose not to use the install script. It should contain all that is needed to create a basic set-up.

4.1.1 The Configuration Module

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

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

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

```
DEBUG = True
```

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

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

sends all errors from the production server to the admin's email address.

```
DATABASES = {
    'default': {
        # Add 'postgresql_psycopg2','postgresql','mysql','sqlite3','oracle'
        'ENGINE': 'django.db.backends.sqlite3',
        # Or path to database file if using sqlite3.
        'NAME': os.path.dirname(os.path.abspath(__file__)) + '/database/local.db',
        'USER': '',                      # Not used with sqlite3.
        'PASSWORD': '',                  # Not used with sqlite3.
        'HOST': '',                      # Set to empty string for localhost.
                                         # Not used with sqlite3.
        'PORT': '',                      # Set to empty string for default.
                                         # Not used with sqlite3.
    }
}
```

```
or

DATABASES = {
    'default': {
            'ENGINE': 'django.db.backends.postgresql_psycopg2',
            'NAME': 'cdr_stats_psql',
            'USER': 'postgresuser',
            'PASSWORD': 'postgrespasswd',
            'HOST': 'localhost',
            'PORT': '5432',
        }
}

or

DATABASES = {
    'default': {
            'ENGINE': 'mysql',
            'NAME': 'cdr_stats_mysql',
            'USER': 'mysqluser',
            'PASSWORD': 'mysqlpasswd',
            'HOST': 'localhost',
            'PORT': '3306',
        }
}

CDR_TABLE_NAME = 'cdr' # Name of the table containing the Asterisk/FreeSwitch CDR

# Only the Asterisk CDR table is supported at the moment,
# but Freeswitch and other platform will be soon
VOIP_PLATFORM = 'asterisk' # asterisk, freeswitch

#MONGODB
#=======
CDR_MONGO_DB_NAME = 'cdr-stats'
CDR_MONGO_HOST = 'localhost'
CDR_MONGO_PORT = 27017
```

sets up the options required for Django to connect to your database.

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

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

```
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 your 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 your project. The custom applications, `cdr` etc. are stored in the project folder along with these custom applications.

Configure different switches

```
#MongoDB(s) to use for import
CDR_MONGO_IMPORT = {
    '127.0.0.1': {
        'db_name': 'cdr-stats',
        'host': 'localhost',
        'port': 27017,
        'collection': 'cdr',
    },
    #'192.168.1.15': {
    #    'db_name': 'freeswitch_cdr',
    #    'host': '192.168.1.15',
    #    'port': 27017,
    #    'collection': 'cdr',
    #},
}
```

4.2 Celery Configuration

4.2.1 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"
```

4.2.2 Launch celery/celerybeat in debug mode

If you don't want to run celeryd and celerybeat as a daemon then

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
```

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

To configure celeryd you will need to tell it where to change directory to, when it starts in order to find your 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
   ```

4.2.4 Troubleshooting

If you can't get the celeryd as a daemon to work, you should try running them in verbose mode:

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

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

CHAPTER

FIVE

DEVELOPER DOC

Contents:

5.1 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/

5.2 Coding Style & Structure

5.2.1 Style

Coding follows the PEP 8 Style Guide for Python Code.

5.2.2 Structure

The CDR-Stats directory:

```
|-- api                 - The code for APIs
|-- cdr                 - The code for CDR
|   `-- fixtures
|-- cdr_alert
|-- static
|   |-- cdr
|   |    |-- css
|   |    |-- js
|   |    |-- icons
|   |    `-- images
|-- resources           - This area is used to hold media files
`-- templates           - This area is used to override templates
```

```
|-- admin
`-- cdr
```

5.3 Objects Description

5.3.1 Switch

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

This defines the Switch

Attributes:

- name - Name of switch.
- ipaddress - ipaddress

Name of DB table: voip_switch

5.3.2 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

5.3.3 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

5.3.4 Alarm

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

This defines the Alarm

Attributes:

- 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

5.3.5 AlertRemovePrefix

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

This defines the Alert Remove Prefix Here you can define the list of prefixes that need to be removed from the dialed digits, imagine all your phone numbers are in the format 5555004432111321 You will need to remove the prefix 5555 in order to analyze the phone numbers

Attributes:

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

Name of DB table: alarm

5.3.6 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

5.3.7 Blacklist

class cdr_alert.models.**Blacklist**(**args*, ***kwargs*)
This defines the Blacklist

Attributes:

- phonenumber_prefix -
- country -

Name of DB table: alert_blacklist

5.3.8 Whitelist

class cdr_alert.models.**Whitelist**(**args*, ***kwargs*)
This defines the Blacklist

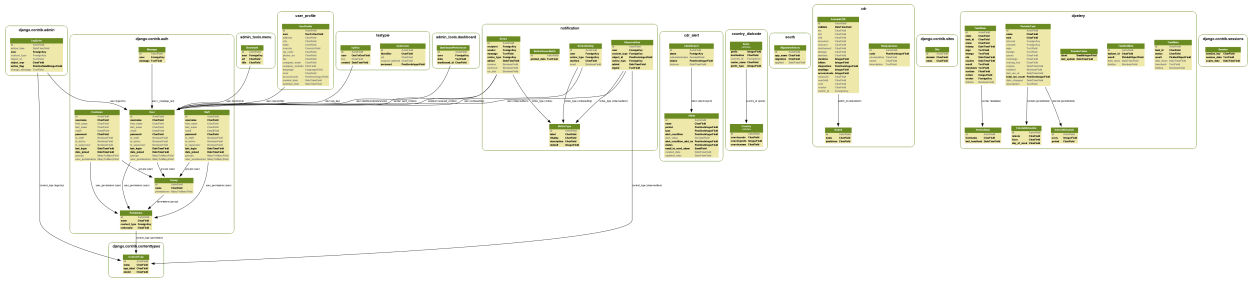
Attributes:

- phonenumber_prefix -
- country -

Name of DB table: alert_whitelist

5.4 Database Design

The current database schema is shown below:



5.5 CDR-Stats Views

5.5.1 index

cdr.views.**index**(*request*)
Index Page of CDR-Stats

Attributes:

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

5.5.2 cdr_view

cdr.views.**cdr_view**(*request*, **args*, ***kwargs*)
List of CDRs

Attributes:

- `template` - cdr/cdr_view.html
- `form` - CdrSearchForm
- `mongodb_data_set` - CDR_MONGO_CDR_COMMON
- `map_reduce` - mapreduce_cdr_view()

Logic Description:

get the call records as well as daily call analytics from mongodb collection according to search parameters

5.5.3 cdr_detail

cdr.views.**cdr_detail**(*request*, **args*, ***kwargs*)
Detail of Call

Attributes:

- `template` - cdr/cdr_detail.html

Logic Description:

get the single call record in detail from mongodb collection

5.5.4 cdr_global_report

cdr.views.**cdr_global_report**(*request*, **args*, ***kwargs*)
CDR global report

Attributes:

- `template` - cdr/cdr_global_report.html
- `form` - SwitchForm
- `mongodb_data_set` - CDR_MONGO_CDR_COMMON
- `map_reduce` - mapreduce_cdr_view()

Logic Description:

get all call records from mongodb collection to create global call report

5.5.5 cdr_dashboard

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

CDR dashboard for a current day

Attributes:

- `template` - cdr/cdr_dashboard.html
- `form` - SwitchForm
- `mongodb_data_set` - CDR_MONGO_CDR_COMMON
- `map_reduce` - mapreduce_cdr_minute_report()

Logic Description:

get all call records from mongodb collection for current day to create hourly report as well as hangup cause/country analytics

5.5.6 cdr_overview

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

CDR graph by hourly/daily/monthly basis

Attributes:

- `template` - cdr/cdr_overview.html.html
- `form` - CdrOverviewForm
- `mongodb_data_set` - CDR_MONGO_CDR_DAILY, CDR_MONGO_CDR_HOURLY
- **`map_reduce` - mapreduce_cdr_hourly_overview() | mapreduce_cdr_monthly_overview()**
  mapreduce_cdr_daily_overview

Logic Description:

get all call records from mongodb collection for all monthly, daily, hourly analytics

5.5.7 cdr_realtime

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

Call realtime view

Attributes:

- `template` - cdr/cdr_realtime.html
- `form` - SwitchForm
- `mongodb_collection` - CDR_MONGO_CONC_CALL_AGG (map-reduce collection)

Logic Description:

get all call records from mongodb collection for concurrent analytics

5.5.8 cdr_graph_by_hour

cdr.views.**cdr_graph_by_hour**(*request*, **args*, ***kwargs*)
CDR graph by hourly basis

Attributes:

- `template` - cdr/cdr_graph_by_hour.html
- `form` - CompareCallSearchForm
- `mongodb_data_set` - CDR_MONGO_CDR_HOURLY
- `map_reduce` - mapreduce_cdr_hour_report()

Logic Description:

get all call records from mongodb collection for hourly analytics for given date

5.5.9 cdr_concurrent_calls

cdr.views.**cdr_concurrent_calls**(*request*, **args*, ***kwargs*)
CDR view of concurrent calls

Attributes:

- `template` - cdr/cdr_graph_concurrent_calls.html
- `form` - ConcurrentCallForm
- `mongodb_data_set` - CDR_MONGO_CONC_CALL_AGG (map-reduce collection)

Logic Description:

get all concurrent call records from mongodb map-reduce collection for current date

5.5.10 customer_detail_change

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

Attributes:

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

Logic Description:

- User is able to change his/her detail.

5.6 CDR-Stats Tasks

5.6.1 sync_cdr_pending

class cdr.tasks.**sync_cdr_pending**
A periodic task that checks for pending calls to import

5.6.2 chk_alarm

class cdr_alert.tasks.**chk_alarm**
A periodic task to determine strange behavior in CDR

Which will get all alarm from system and checked with alert condition value & if it is matched, user will be notified via mail

Usage:

chk_alarm.delay()

5.6.3 blacklist_whitelist_notification

class cdr_alert.tasks.**blacklist_whitelist_notification**
Send notification to user while destination number matched with blacklist or whitelist

Usage:

blacklist_whitelist_notification.delay(notice_type)

5.6.4 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()

5.7 Test Case Descriptions

5.7.1 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
```

5.7.2 How to run test

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
```

CHAPTER

SIX

API REFERENCE

Contents:

6.1 SwitchResource

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

Attributes Details:

- name - Name of switch.
- ipaddress - ipaddress

Read:

CURL Usage:

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

6.2 HangupCauseResource

class api.resources.**HangupCauseResource**(*api_name=None*)

Attributes Details:

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

Read:

CURL Usage:

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

6.3 CdrDailyResource

class api.resources.**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
    }
]
```

6.4 CdrResource

class `api.resources.`**`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"
}
```

CHAPTER

SEVEN

CONTRIBUTING

7.1 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.

7.1.1 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.

7.1.2 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.

7.1.3 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.

7.1.4 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.

7.1.5 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.

7.1.6 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.

7.2 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.

7.2.1 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

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

7.3 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

EIGHT

FREQUENTLY ASKED QUESTIONS

8.1 General

8.1.1 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.

8.1.2 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.

8.1.3 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

NINE

TROUBLESHOOTING

9.1 Where to find help

9.1.1 Documentation:

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

9.1.2 Mailing list:

We have set up a mailing list at http://groups.google.com/group/cdr-stats

9.1.3 Forum:

We have a forum at http://forum.cdr-stats.org/

9.1.4 Support:

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

9.2 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

9.3 Run in debug mode

Make sure you stop the services 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
```

9.4 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 you probably want to run the monitor 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, you can use the help command:

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

Now you can visit the webserver celerymon starts by going to: http://localhost:8989

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

Project documentation is hosted on CDR-Stats website : http://cdr-stats.readthedocs.org/

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

10.4 Contributing

Development of *CDR-Stats* happens at Github : https://github.com/Star2Billing/cdr-stats

You are highly encouraged to participate in the development of *CDR-Stats*. If you would prefer not to use Github, you are welcome to send us regular patches

Be sure to also read the *Contributing* section in the documentation.

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

INDICES AND TABLES

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

PYTHON MODULE INDEX

INDEX