Qualcomm Technologies, Inc.

Genuine Device Detection System 1.0.0

API Installation Guide

GDDS-Installation-Guide-API-1.0.0

November 10, 2020

Revision history

Revision	Date	Description
A	November 2020	Initial release

Contents

1. Introduction

1.1 Purpose & Scope

This document provides:

- Installation instructions for the Genuine Device Detection Subsystem
- Instructions for running test commands to verify GDDS API installation

1.2 Definitions, Acronyms & Abbreviations

Table 1 - Definitions, Acronyms & Abbreviations

Term	Explanation
DIRBS	Device Identification, Registration & Blocking System
GDDS	Genuine Device Detection Subsystem
OS	Operating System
PostgreSQL	PostgreSQL open source object-relational database system
Nginx	An open source, lightweight, high-performance web server or proxy server
uWSGI	uWSGI is used for serving Python web applications

1.3 References

N.A

1.4 Getting Started

The instructions provided in this document assume that the required equipment (hardware, software) has been installed and configured with Ubuntu 16.04. Refer to the Ubuntu Installation Guide for additional installation help.

The installer should be familiar with Linux command line.

2. Installation

NOTE: The reader acknowledges and agrees that he is entirely and solely responsible for the selection and use of all third-party software modules downloaded and installed by this installation method, including securing all appropriate and proper rights of use to any of such third-party software modules and to comply fully with any terms of use that may apply to or accompany any such third-party software modules.

2.1 System Requirements

2.1.1 Software Requirements

- Python 3.6
- Ubuntu 16.0
- PostgreSQL 10
- Nginx 1.14.X
- uWSGI 2.0

2.1.2 Hardware Requirements

Minimum hardware requirements

- At least 1 GB of RAM
- At least 8 GB of disk space

2.1.3 Operating System

This subsystem will be installed and configured with Ubuntu 16.04. Refer to the Ubuntu Installation Guide for additional installation help.

- Ubuntu 16.04
- non-root user

You should have a regular, non-root user account on your server with sudo privileges (in this installation guide the user is referred to as 'gdds-user')

2.1.4 Database Support

NOTE: Creating a new database from scratch assumes that you are already running a PostgreSQL instance.

A complete guide for PostgreSQL installation and configuration can be found on PostgreSQL website

2.2 Extracting Software Release

The GDDS software release can be downloaded via one of the following two methods. To extract the contents of the distribution, run either:

Method 1: Download and unzip from GitHub

```
unzip Genuine-Device-Detection-Subsystem-master.zip
```

Method 2: Clone the repository from GitHub

```
git clone https://github.com/CACF/Genuine-Device-Detection-
Subsystem.git
```

Copy the content to the user home directory e.g. /home/gdds-user (you may have different home directory according to user)

2.3 Manual Installation

- Ensure the APT package index is updated
  ```
  sudo apt-get update --fix-missing
  ```
- Install basic required packages
  ```
  sudo apt-get install nginx git python3 virtualenv libpython3-dev
  python3-pip python3-dev postgresql
  ```
- Go to path /home/gdds-user/Genuine-Device-Detection-Subsystem
  ```
  cd /home/gdds-user/Genuine-Device-Detection-Subsystem
  ```
- Create virtual environment and activate it
  ```
  virtualenv –p python3 venv
  source venv/bin/activate
  ```
- Install all libraries from requirements.txt in virtual
  ```
  pip3 install -r requirements.txt
  ```
- Nginx does not support python application so we need to install uWSGI to run python application through Nginx, below is the command to install uWSGI
  ```
  pip3 install uwsgi
  ```
- Deactivate the virtual environment:
  ```
  deactivate
  ```

3. Configuration

3.1 Nginx Configuration

Remove Nginx default configuration and create new configuration file for the GDDS app

```
sudo rm /etc/nginx/sites-enabled/default
```

- Now create a new configuration file in the root path

```
nano /home/gdds-user/Genuine-Device-Detection-Subsystem/gdds-
  nginx.conf
```

- Copy the below lines

```
server {
listen      80;
server_name localhost;
charset     utf-8;
client_max_body_size 75M;
location / {try_files $uri @dirbs-gdds;}
location @dirbs-gdds
{
include uwsgi_params;
uwsgi_pass unix: /home/gdds-user/Genuine-Device-Detection-Subsystem/uwsgi.sock;
  }
}
```

- Symlink the new created file to Nginx's configuration files directory and restart Nginx

```
sudo ln -s /home/gdds-user/Genuine-Device-Detection-Subsystem/gdds-nginx.conf
/etc/nginx/conf.d/
```

- Verify nginx configuration

```
sudo nginx –t
```

- Restart Nginx Service

```
sudo service nginx restart
```

3.2 uWSGI Configuration

- Create a new configuration file in the root path and copy the below lines

```
nano /home/gdds-user/Genuine-Device-Detection-Subsystem/uwsgi.ini
```

- Add below lines in this configuration file:

```
[uwsgi]
#application's base folder
base = /home/gdds-user/Genuine-Device-Detection-Subsystem/

#python module to import
module = app
chdir = %(base)
home = %(base)/venv
pythonpath = %(base)

master = true
processes = 10
cheaper = 2
cheaper-initial = 5
cheaper-step = 1
cheaper-algo = spare
cheaper-overload = 5

enable-threads = true
max-requests = 500
max-worker-lifetime = 120

ignore-sigpipe=true
ignore-write-errors=true
disable-write-exception=true

#socket file's location
socket = /home/gdds-user/Genuine-Device-Detection-Subsystem/%n.sock

#permissions for the socket file
chmod-socket = 666
chown-socket = gdds-user:gdds-user

#ownership of uwsgi service
uid = gdds-user
gid = gdds-user

#the variable that holds a flask application inside the module imported at line #6
callable = app

#location of log files
logto = /var/log/uwsgi/%n.log
```

- Create a directory vassals in /etc/uwsgi/
  ```
  sudo mkdir -p /etc/uwsgi/vassals
  ```

- Create Symlink in this directory to uwsgi ini config file
  ```
  sudo ln -s /home/gdds-user/Genuine-Device-Detection-Subsystem/uwsgi.ini \
  /etc/uwsgi/vassals/uwsgi.ini
  ```

- Create a new directory for log files
  ```
  sudo mkdir -p /var/log/uwsgi
  ```

- Change ownership of logs directory to gdds-user
  `sudo chown -R gdds-user:gdds-user /var/log/uwsgi/`

3.3 uWSGI Service Configuration

Configure the uwsgi to run as a service on the server.

- Create an init script at location
  `sudo nano /etc/systemd/system/uwsgi.service`
- Copy below lines in to the script file

```
[Unit]
Description=uWSGI Emperor service
After=syslog.target

[Service]
ExecStart=/home/gdds-user/Genuine-Device-Detection-Subsystem/venv/bin/uwsgi --
emperor /etc/uwsgi/vassals/
Restart=always
KillSignal=SIGQUIT
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

- Reload system defaults to update the script in system services
  `sudo systemctl daemon-reload`
- Start uwsgi to start the application
  `sudo service uwsgi start`
- Go to the web-browser and enter the URL of the server to check that the service is running

3.4 GDDS Configuration and Initialization

To configure file according to database server and credentials, edit config.yml in directory /home/gdds-user/Genuine-Device-Detection-Subsystem/

Change the hostname, port, username, password and database name as per requirements.
Within the same directoy of GDDS root, activate the virtual environment.

To activate the virtual environment created earlier at `/home/gdds-user/Genuine-Device-Detection-Subsystem` and start database initialization.

```
cd /home/gdds-user/Genuine-Device-Detection-Subsystem
```

```
source venv/bin/activate
```

Follow below steps for database initialization and migration:

- Run Database migrations using:

  ```
  python manage.py db init
  python manage.py db migrate
  python manage.py db upgrade
  ```

This will automatically create and migrate database schemas and requirements.

- Install setup for default configurations using:

  ```
  python setup.py install
  ```

- Start GDDS development server using:

  ```
  python run.py
  ```

This will start a flask development environment.

- To run unit tests, run:

  ```
  pytest -v -s
  ```

3.4.1 CLI Commands

Run below CLI commands in Venv as per requirements

- To load GSMA-TAC-Database File

  ```
  update-gsma-tacs <file_path>
  e.g.
  update-gsma-tacs /home/gdds_files/gsma_tac_file.csv
  ```

- To load Duplication-List from DIRBS-CORE

  ```
  import duplication_list <file_path>
  e.g.
  import duplication_list /home/gdds_files/dup_list.csv
  ```

- Send Request-SMS to all duplicated users to ask for device info

  ```
  send-request-sms all
  ```

- Send Request-SMS only to unnotified users to ask for device info

  ```
  send-request-sms unnotified
  ```

- To run Comparison-Algorithm

  ```
  run-comparison
  ```

- To send SMS to all Genuine marked users about their Device pairing

  ```
  send-intimation-sms genuine
  ```

- To send SMS to all Duplicated marked users about their Device black-listing

  ```
  send-intimation-sms duplicated
  ```

- To create Black-List & Pairing-List for Authority

  ```
  generate-lists
  ```

3.4.2 Other Helpful Commands

- To Upgrade already installed database:

  ```
  python manage.py db upgrade
  ```

- To run unit and regression tests:

  ```
  pytest -v -s
  ```

- To enable different languages & activate their translations from English

  ```
  pybabel extract -F babel.cfg -k _l -o messages.pot .
  pybabel init -i messages.pot -d app/translations -l <language-code>
  ```

  e.g to translate in Spanish

  ```
  pybabel init -i messages.pot -d app/translations -l es
  ```

  Finally, to compile the language

  ```
  pybabel compile -d app/translations
  ```

  To update any translation after compilation

  ```
  pybabel update -i messages.pot -d app/translations -l es
  ```

- Restart uWSGI service

  ```
  sudo service uwsgi restart
  ```

4. Testing

- To check nginx configuration for errors
  ```
  sudo nginx –t
  ```

- To get detailed logs of uWSGI service. uWSGI can be run without service command in foreground
  ```
  uwsgi --ini /home/gdds-user/Genuine-Device-Detection-
  Subsystem/uwsgi.ini
  ```