Device Pairing Subsystem 3.0.0

API Installation Guide

DPS-Installation-Guide-API-3.0.0

November 09, 2020

**Revision history**

| **Revision** | **Date** | **Description** |
| --- | --- | --- |
| A |  | Initial release |
| B | November 2020 | Release 3.0.0 |

Contents

[1. Introduction 4](#_Toc57201230)

[1.1 Purpose & Scope 4](#_Toc57201231)

[1.2 Definitions, Acronyms & Abbreviations 4](#_Toc57201232)

[1.3 References 4](#_Toc57201233)

[1.4 Getting Started 4](#_Toc57201234)

[2. Installation 5](#_Toc57201235)

[2.1 System Requirements 5](#_Toc57201236)

[2.1.1 Software Requirements 5](#_Toc57201237)

[2.1.2 Hardware Requirements 5](#_Toc57201238)

[2.1.3 Operating System 5](#_Toc57201239)

[2.1.4 Database Support 5](#_Toc57201240)

[2.2 Extracting Software Release 6](#_Toc57201241)

[2.3 Manual Installation 6](#_Toc57201242)

[3. Configuration 7](#_Toc57201243)

[3.1 Nginx Configuration 7](#_Toc57201244)

[3.2 uWSGI Configuration 7](#_Toc57201245)

[3.3 uWSGI Service Configuration 8](#_Toc57201246)

[3.4 DPS Configuration and Initialization 9](#_Toc57201247)

[3.4.1 Other Helpful Commands 10](#_Toc57201248)

[3.5 Configuration File Settings 11](#_Toc57201249)

[4. Testing 13](#_Toc57201250)

Tables

[Table 1 - Definitions, Acronyms & Abbreviations 4](#_Toc56257154)

# **Introduction**

# **Purpose & Scope**

This document provides:

* Installation instructions for the Device Pairing Subsystem (DPS)
* Instructions for running test commands to verify DPS API installation

# **Definitions, Acronyms & Abbreviations**

Table 1 - Definitions, Acronyms & Abbreviations

| Term | Explanation |
| --- | --- |
| DIRBS | Device Identification, Registration & Blocking System |
| DPS | Device Pairing 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 |
| API | Application Program Interface |

# **References**

N.A

# **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](https://help.ubuntu.com/lts/installation-guide/i386/install.en.pdf) for additional installation help.

The installer should be familiar with Linux command line.

# **Installation**

1. 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.

Qualcomm Technologies, Inc. does not undertake any obligations, duties, or other responsibilities in connection with the selection or use by the reader of any of such third-party software modules.

## System Requirements

### Software Requirements

* Python 3.X
* PostgreSQL 10
* Nginx 1.14.X
* uWSGI 2.0

### Hardware Requirements

Minimum hardware requirements

* At least 1 GB of RAM
* At least 8G of disk space

### Operating System

This subsystem will be installed and configured with Ubuntu 16.04. Refer to the [Ubuntu Installation Guide](https://help.ubuntu.com/lts/installation-guide/i386/install.en.pdf) 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 ‘dps-user’)

### Database Support

1. 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](https://www.postgresql.org/)

## Extracting Software Release

The DPS 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 Device-Pairing-Subsystem.zip

Method 2: Clone the repository from GitHub  
git clone https://github.com/CACF/Device-Pairing-Subsystem.git

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

## 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/dps-user/Device-Pairing-Subsystem

cd /home/dps-user/Device-Pairing-Subsystem

* Create virtual environment and activate it

virtualenv –m python3 venv

source venv/bin/activate

* Install all libraries from requirements.txt in virtual environment

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

# **Configuration**

## Nginx Configuration

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

rm /etc/nginx/sites-enabled/default

* Now create a new configuration file in the root path

nano /home/dps-user/Device-Pairing-Subsystem/dps-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-dps;}  location @dirbs-dps  {  include uwsgi\_params;  uwsgi\_pass unix:/home/dps-user/Device-Pairing-Subsystem/uwsgi.sock;  }  } |

* Symlink the new created file to Nginx’s configuration files directory and restart Nginx

sudo ln -s /home/dps-user/Device-Pairing-Subsystem/dps-nginx.conf \

/etc/nginx/conf.d

* Verify nginx configuration

sudo nginx –t

* Restart Nginx Service

sudo service nginx restart

## uWSGI Configuration

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

nano /home/dps-user/Device-Pairing-Subsystem/uwsgi.ini

* Add below lines in this configuration file:

|  |
| --- |
| [uwsgi]  #application's base folder  base = /home/dps-user/Device-Pairing-Subsystem  #python module to import  app = run  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  #socket file's location  socket = /home/dps-user/Device-Pairing-Subsystem/%n.sock  #permissions for the socket file  chmod-socket = 666  chown-socket = dps-user:dps-user  #ownership of uwsgi service  uid = dps-user  gid = dps-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/dps-user/Device-Pairing-Subsystem/uwsgi.ini \ /etc/uwsgi/vassals/uwsgi.ini

* Create a new directory for log files

sudo mkdir -p /var/log/uwsgi

* Change ownership of the logs directory to dps-user

chown -R dps-user:dps-user /var/log/uwsgi/

## uWSGI Service Configuration

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

* Create an init script at location

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/dps-user/Device-Pairing-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](http://dps-server-ip/) of the server to check that the service is running

## DPS Configuration and Initialization

To configure file according to database server and credentials, edit config.yml file in directory /home/dps-user/Device-Pairing-Subsystem/etc

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

To activate the virtual environment created earlier at /home/dps-user/Device-Pairing-Subsystem and start database initialization

cd /home/dps-user/Device-Pairing-Subsystem

source venv/bin/activate

* Create folder “Downloads” into the app directory.

mkdir/home/dps-user/Device-Pairing-Subsystem/Downloads

* Install all libraries from requirements.txt in virtual environment

pip3 install -r requirements.txt

* Create virtual environment using **virtualenv** and activate it:

virtualenv venv

source venv/bin/activate

Make sure that virtual-env is made using Python3 and all the required dependencies are installed.

* Run Database migrations using:

python manage.py db init

python manage.py db migrate

python manage.py db upgrade

python manage.py create\_view

This will automatically create and migrate database schemas and requirements.

* Start DPS development server using:

python run.py

This will start a flask development environment for DPS.

* To run unit tests, run:

pytest -v -s

### ****Other Helpful Commands****

* To Upgrade already installed database:

python manage.py db upgrade

* To generate full pairing-list for DIRBS Core:

python scripts/pairlist\_gen\_complete.py

* To generate delta pairing-list for DIRBS Core:

python scripts/pairlist\_gen\_delta.py

* To delete un-confirmed pairs to clean-up main DB:

python scripts/unconfirmed\_pair\_deletion.py

* 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

## Configuration File Settings

DPS core configuration paramaters will be defined in configuration file “config.yml” and will be placed at root directory. The details of the parameters, their default values and usage are described below;

pair\_limit: Defines how many secondary pairs you want DPS to allow. The default value is set to 4 and can be changed acoording to business needs.

pc\_length: Controls the length of pair-code. Pair-Code acts like OTP (One-Time-Password) and will be given to subscriber upon registering the device with authority. The default value is set to 8.

imeis\_per\_device: As the name suggests, this parameter allows the total number of IMEIs that can be registered against single device. The default value is set to 5 but can be altered as per business needs.

**Database connection Parameters**

These DB parameters must be set before initialization of application according to your database settings

Dbname: The parameter must be set with database name. Default is the dummy database name.

Dbusername: Database username must be provided in this parameter. Default is the dummy username.

Dbpassword: Database password must be provided in this parameter. Default is the dummy password.

Dbhost: The IP address of database will be mentioned here. Default value is set to localhost.

**Database tunning Parameters**

These parameters are already set for optimized performance of DB but can be modified as per needs

pool\_size: The parameter controls the number of sessions in pool for DB. Its default value is 100

pool\_timeout: Default value of this parameter is 20

pool\_recycle: Default value of this parameter is 10

overflow\_size: overflow size is kept to 275

**Operator related parameters**

Just like DB, the values of these parameters must also be set as per opertors’ requirements before running the app.

num\_of\_mnos: 5

MNO\_Names: ["jazz", "telenor", "ufone", "zong", "warid"]

To add the sixth operator, one should simply add below line

MNO\_6: 'Vodafone'

Just like DB parameters, following Kannel parameters must also be added

kannel\_sms: 'http://192.168.100.70:13013/cgi-bin/sendsms'

kannel\_username: 'tester'

kannel\_password: 'foobar'

kannel\_smsc: 'at'

kannel\_shortcode: '7787'

To add multilanguage support

supported\_languages:

languages: ['es', 'id', 'en']

default\_language: 'en'

kannel\_sms: This parameter contains the URL configured for Kannel server. Default value is the dummy server address.

pairlist\_path: The parameter contains the path at which pair-lists will be saved. One of the products of DPS is pairing-list which is required by DIRBS-CORE to generate exception lists. You need to mention the directory, in this parameter, at which you want to save these pair-lists

Download\_Path: The path for storing bulk MSISDN-files for operators. DPS will save these files in that directory which need manual housekeeping later on.

Upload\_Path: The MSISDN-IMSI pairing files which are uploaded by operators will be placed on a directory mentioned in this parameter.

sms\_pair\_limit\_breached: This is a sample parameter for one of the responses provided by DPS to Kannel. You can set as many parameters as possible for different kind of SMS(s). The value in the parameter will be provided to Kannel as string.

# **Testing**

* To test Nginx server configuration, run below mentioned command:

nginx –t

* To get detailed logs of uWSGI service. uWSGI can be run without service command in foreground

uwsgi --ini /home/dps-user/Device-Pairing-Subsystem/uwsgi.ini