Device Verification Subsystem 1.0.0

API Installation Guide

DVS-API-Installation-Guide-1.0.0

July 12, 2018

**Revision history**

| **Revision** | **Date** | **Description** |
| --- | --- | --- |
| A |  | Initial release |

Contents

[1 Introduction 4](#_Toc527033785)

[1.1 Purpose 4](#_Toc527033786)

[1.2 Definitions, Acronyms & Abbreviations 4](#_Toc527033787)

[1.3 References 4](#_Toc527033788)

[1.4 Getting Started 4](#_Toc527033789)

[2 Installation 5](#_Toc527033790)

[2.1 System Requirements 5](#_Toc527033791)

[2.1.1 Software Requirements 5](#_Toc527033792)

[2.1.2 Hardware Requirements 5](#_Toc527033793)

[2.1.3 Operating System 5](#_Toc527033794)

[2.2 Extracting Software Release 5](#_Toc527033795)

[2.3 Manual Installation 6](#_Toc527033796)

[3 Configuration 8](#_Toc527033797)

[3.1 Nginx Configuration 8](#_Toc527033798)

[3.2 uWSGI Configuration 9](#_Toc527033799)

[3.3 uWSGI Service Configuration 10](#_Toc527033800)

[4 Testing 11](#_Toc527033801)

**Tables**

[Table 1- Definitions, Acronyms & Abbreviations 4](#_Toc527623053)

# Introduction

## Purpose

This document provides:

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

## Definitions, Acronyms & Abbreviations

**Table 1- Definitions, Acronyms & Abbreviations**

| Term | Explanation |
| --- | --- |
| DIRBS | Device Identification, Registration & Blocking System |
| DVS | Device Verification Subsystem |
| OS | Operating System |
| Nginx | An open source, lightweight, high-performance web server or proxy server. |
| uWSGI | The uWSGI project aims at developing a full stack for building hosting services |
| 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
* Ubuntu 16.0
* Nginx 1.14.X
* uWSGI 2.0

### Hardware Requirements

Minimum hardware requirements:

* At least 512 MB of RAM
* At least 1G of disk space

### Operating System

This system 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.

## Extracting Software Release

The DVS software release is distributed as a tar.gz file. To extract the contents of the  
distribution, run:  
tar xvzf dirbs-dvs-api-1.0.0.tar.gz

Copy the contents to the web root directory e.g. /var/www/html (default Nginx web root directory)

## Manual Installation

* Ensure the APT package index is updated

apt-get update --fix-missing

* Install basic required packages

apt-get install nginx git python3 python3-pip python3-dev

libpython3-dev virtualenv

* Go to path /var/www/html/dirbs-dvs-api-1.0.0

pushd /var/www/html/dirbs-dvs-api-1.0.0

* Create virtual environment install requirements

virtualenv –p python3 venv

source venv/bin/activate

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

* Install RabbitMQ

apt update && sudo apt upgrade

apt-get install rabbitmq-server

systemctl enable rabbitmq-server

systemctl start rabbitmq-server

* Install Celery in python virtual environment
* Source into already created virtual environment

source /var/www/html/dirbs-dvs-api-1.0.0/venv/bin/activate

pip3 install celery

* Start the Workers as Daemons so that they are started automatically at server startup
* Create a new service definition file in /etc/systemd/system/celeryd.service. Change the “User” and “Group” properties according to your actual user and group name
* For our setup we have created a user celery using below command

adduser celery

|  |
| --- |
| [Unit]  Description=Celery Service  After=network.target  [Service]  Type=forking  User=**celery**  Group=**celery**  EnvironmentFile=/etc/default/celeryd  WorkingDirectory=/var/www/html/dirbs-dvs-api-1.0.0/  ExecStart=/bin/sh -c '${CELERY\_BIN} multi start ${CELERYD\_NODES} -B\  -A ${CELERY\_APP} --pidfile=${CELERYD\_PID\_FILE} \  --logfile=${CELERYD\_LOG\_FILE} --loglevel=${CELERYD\_LOG\_LEVEL} ${CELERYD\_OPTS}'  ExecStop=/bin/sh -c '${CELERY\_BIN} multi stopwait ${CELERYD\_NODES} \  --pidfile=${CELERYD\_PID\_FILE}'  ExecReload=/bin/sh -c '${CELERY\_BIN} multi restart ${CELERYD\_NODES} -B \  -A ${CELERY\_APP} --pidfile=${CELERYD\_PID\_FILE} \  --logfile=${CELERYD\_LOG\_FILE} --loglevel=${CELERYD\_LOG\_LEVEL} ${CELERYD\_OPTS}'  [Install]  WantedBy=multi-user.target |

* Create a configuration file “celeryd” in /etc/default/ directory

|  |
| --- |
| #The name of the workers. This example will create two workers  CELERYD\_NODES=”worker1 worker2”  # The name of the Celery App, should be the same as the python file  # where the Celery tasks are defined  CELERY\_APP=”app.celery”  # log and PID directories  CELERYD\_LOG\_FILE=”/var/log/celery/%n%I.log”  CELERYD\_PID\_FILE=”/var/run/celery/%n.pid”  #log level  CELERYD\_LOG\_LEVEL=INFO  #Path to celery binary, that is in your virtual environment  CELERY\_BIN=/var/www/html/dirbs-dvs-api-1.0.0/venv/bin/celery |

* Create log and pid directories

mkdir /var/log/celery /var/run/celery/

chown celery:celery /var/log/celery /var/run/celery

* Reload systemctl daemon. You should run this command each time you make any change in the service definition file.

systemctl daemon-reload

* Enable the service to startup at boot

systemctl enable celeryd

* Start the service

systemctl start celeryd

# Configuration

## Nginx Configuration

Remove Nginx default configurations and create new configuration file for the DVS app

rm /etc/nginx/sites-enabled/default

* Now create a new configuration file n the root path

nano /var/www/html/dirbs-dvs-api-1.0.0/dvs.conf

* Copy the below lines

|  |
| --- |
| server {  listen 80;  server\_name localhost;  charset utf-8;  client\_max\_body\_size 75M;  location / {try\_files $uri @dirbs-dvs-api-1.0.0;}  location @dirbs-dvs-api-1.0.0  {  include uwsgi\_params;  uwsgi\_pass unix:/var/www/html/dirbs-dvs-api-1.0.0/uwsgi.sock;  }  } |

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

ln –s /var/www/html/dirbs-dvs-api-1.0.0/dvs.conf /etc/nginx/conf.d/

* Verify Nginx configuration

nginx –t

* Restart Nginx Service

service nginx restart

## uWSGI Configuration

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

nano /var/www/html/dirbs-dvs-api-1.0.0/uwsgi.ini

* Add below lines in this configuration file:

|  |
| --- |
| [uwsgi]  #application's base folder  base = /var/www/html/dirbs-dvs-api-1.0.0/  #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 = /var/www/html/dirbs-dvs-api-1.0.0/%n.sock  #permissions for the socket file  chmod-socket = 666  chown-socket = www-data:www-data  #ownership of uwsgi service  uid = www-data  gid = www-data  #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/

mkdir –p /etc/uwsgi/vassals

* Create symlink in that directory to uwsgi ini config file

ln -s /var/www/html/dirbs-dvs-api-1.0.0/uwsgi.ini \

/etc/uwsgi/vassals/uwsgi.ini

* Create a new directory for log files

mkdir –p /var/log/uwsgi

* Change ownership of the web root directory and logs directory to the web-user

chown -R www-data:www-data /var/www/html/dirbs-dvs-api-1.0.0/

chown -R www-data:www-data /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 the script file

|  |
| --- |
| [Unit]  Description=uWSGI Emperor service  After=syslog.target  [Service]  ExecStart=/var/www/html/dirbs-dvs-api-1.0.0/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

systemctl daemon-reload

* Start uwsgi to start the application

service uwsgi start

* Go to the web-browser and enter the [URL](http://dvs-server-ip/) of the server to check the service running

# 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 /var/www/dirbs-dvs-api-1.0.0/uwsgi.ini