Installing a FreeDMR server (non Docker version) as well as its MONITOR

Document v1.5.1 realized by **Patrick F1FQN** (**Happy buddha**) member of the Shield Group
as of December 17, 2021

FreeDMR: http://www.freedmr.uk/

Lot of thanks to all FreeDMR coders team

TheShield server: http://theshield.site:8080

TheGate server (La French Connection)
http://thegate.hblink.net:8080

Thanks for their collaboration:

Nicolas **F4ISE** (Shield Group)
Youness **CN8VX (**DMR Morocco **)**
Daniel **F1RMB** (Developer of the Pistar fork and the GD77 alternative firmware team)

Features of a FREEDMR server

- **Dial-a-TG** - This function allows you to call any TG number by private call by having it routed by the TG9 (similar to the old reflector system: On your transceiver, connection to the TG9 + TG number)
- **Voice ID** : Text to speech announcing on which server you are connected.
- **Dynamic bridging** : In OPENBRIDGE mode, no rule is to be written in the rules.py file.
- **Dynamic static bridging** : Automatic transfer of traffic to all other OPENBRIDGES).
- **Single port operation for hotspot connectivity** : (a single sign-on port is required: 62031 for all your guest hotspots
- **Optimized management for OPENBRIDGE** duplicate, missing and / or out of service packets.
- **Single mode** : When enabled, only one TG can be active on a TS at a time.
- **Timer control** for user-activated newsgroups.
- **Support for client configuration** sent via the Homebrew Protocol Options line The OPTIONS line can be programmed in your PISTAR or other Hotspot or in your DMR applications (Droidstar, DudeStar, etc.)
- **Recognition of HBP DMRA packets**
- **Support for international languages** for voice prompts for Dial-a-TG
- **Multiple open bridges and no loops** - makes possible a mesh network without disturbances.
- **Detection and handling of duplicate and failed packets**.
- **Loop detection and management**.
- **Improved OpenBridge / Bridge control protocol**
- **OpenBridge works behind NAT routing and with dynamic IP**
- **Modeled systems generator** (Allows you to automatically generate a certain number of Hotspots zones.
- **Docker image for a preconfigured installation** but not really documented. To be reserved for Docker specialists

1- Basic server installation

Read and understand before you start creating a FreeDMR server:

You have the possibility of creating an independent and private server but also a server integrating an existing network. This document gives you the technical bases to achieve it. However, if you wish to create a FreeDMR server whose purpose is to integrate a network belonging to one or more organizations, there will necessarily be naming and connection rules to be observed.
The purpose of this document is therefore only for educational purposes and oriented solely with the aim of creating a private server.

On Debian 10 or higher

After installing and securing a freshly installed Debian distribution:

sudo apt update
sudo apt upgrade
sudo apt install git sudo apt install python3-pip sudo apt install python-pip **(useless in Debian 11)**
python3 -m pip install pip --upgrade cd / opt **(useless in Debian 11)**

git clone https://gitlab.hacknix.net/hacknix/FreeDMR.git

This command imports the installation directory locally on your server. Following the launch of this one, a FreeDMR directory will be created under**/opt** While remaining at the level of **/opt**

cd / opt / FreeDMR
chmod + x install.sh
sudo ./install.sh
cp FreeDMR-SAMPLE.cfg hblink.cfg **(here, we copy the example file into a configuration file)**
cp rules_SAMPLE.py rules.py (ditto here for the TG configuration file)

Attention, here the first trap set by the developers: FreeDMR-SAMPLE is a "-" and rules_SAMPLE, it is a "_"

This done, our 2 configuration files **hblink.cfg** and **rules.py** are created but not yet able to make your server operational.

We will come back to how to create your own TG (s) but for the moment and to illustrate the basic configuration, we will connect your server to a partner server in order to retrieve the TGs and make them available on yours.

2- The 3 ways to connect your server to a partner server

- PEER mode
- XLXPEER mode
- OPENBRIDGE mode

- **PEER mode:** Your server connects like a Pistar-like Hotspot and "retrieves" one or more TGs from a server **partner (HBLink, FreeDMR, IPSC2, TGIF)**

- **XLXPEER mode:** Your server connects like a Pistar-like Hotspot and “retrieves” one or more TGs from a gateway server **XLX type**

- **OPENBRIDGE mode** is a dedicated communication channel between 2 servers and has no limit as to the number of TGs. It is then necessary to contact the SYSOP of the remote server in order to agree and exchange the IP addresses, connection ports and passwords.**(HBLink, FreeDMR, IPSC2, Brandmeister)**

First, we are going to make some adaptations to our 2 previously created files (hblink.cfg and rules.py) in order to **connect us in PEER mode** on our THEGATE server and thus retrieve the **TG9379**. Let's go !!

Let's open our hblink.cfg file (connection management file) which should appear by default like this:

nano /opt/FreeDMR/hblink.cfg

```
 [GLOBAL]
PATH: ./
PING_TIME: 10
MAX_MISSED: 3
USE_ACL: True
REG_ACL: PERMIT: ALL
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
GEN_STAT_BRIDGES: True
ALLOW_NULL_PASSPHRASE: True
ANNOUNCEMENT_LANGUAGES: en_GB, en_US, es_ES, fr_FR, de_DE, dk_DK, it_IT, no_NO, pl_PL, se_SE, pt_PT, cy_GB, el_GR, CW
SERVER_ID: 0000

[REPORTS]
REPORT: True
REPORT_INTERVAL: 60
REPORT_PORT: 4321
REPORT_CLIENTS: 127.0.0.1

[LOGGER]
LOG_FILE: freedmr.log
LOG_HANDLERS: file-timed
LOG_LEVEL: INFO
LOG_NAME: FreeDMR

[ALIASES]
TRY_DOWNLOAD: True
PATH: ./
PEER_FILE: peer_ids.json
SUBSCRIBER_FILE: subscriber_ids.json
TGID_FILE: talkgroup_ids.json
PEER_URL: https://www.radioid.net/static/rptrs.json SUBSCRIBER_URL:
https://www.radioid.net/static/users.json TGID_URL: http://
downloads.freedmr.uk/downloads/talkgroup_ids.json STALE_DAYS: 7


[MYSQL]
USE_MYSQL: False
USER: hblink
PASS: mypassword
DB: hblink
SERVER: 127.0.0.1
PORT: 3306
TABLE: repeaters

[OBP-TEST]
MODE: OPENBRIDGE
ENABLED: False
IP:
PORT: 62044
NETWORK_ID: 1
PASSPHRASE: mypass
TARGET_IP:
TARGET_PORT: 62044
USE_ACL: True
SUB_ACL: DENY: 1
TGID_ACL: PERMIT: ALL
RELAX_CHECKS: True
ENHANCED_OBP: True

[SYSTEM]
MODE: MASTER
ENABLED: True
REPEAT: True
MAX_PEERS: 1
EXPORT_AMBE: False
IP: 127.0.0.1
PORT: 54000
PASSPHRASE:
GROUP_HANGTIME: 5
USE_ACL: True
REG_ACL: DENY: 1
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
DEFAULT_UA_TIMER: 60
SINGLE_MODE: True
VOICE_IDENT: True
TS1_STATIC:
TS2_STATIC:
DEFAULT_REFLECTOR: 0
ANNOUNCEMENT_LANGUAGE: en_FR
GENERATOR: 100
```

3-Creation of a PEER section to a server (Example, the THEGATE server)

[**THEGATE**]
MODE: PEER
ENABLED: **True**
LOOSE: False
EXPORT_AMBE: False
IP:
PORT: 54301
MASTER_IP: **thegate.hblink.net**
MASTER_PORT**: 62031**
PASSPHRASE: **passw0rd**
CALLSIGN: **Your callsign**
RADIO_ID: **Your DMR Id CCS7 or CCS7 + suffix from 01 to 99**
RX_FREQ: 449000000
TX_FREQ: 444000000
TX_POWER: 25
COLORCODE: 1
SLOTS: 2
LATITUDE: 46.764043
LONGITUDE: 5.835659
HEIGHT: 320
RENTAL: **Your City, Country or QRA locator**
DESCRIPTION: Virtual hotspot URL:
www.google.fr
SOFTWARE_ID: 20170620
PACKAGE_ID: MMDVM_HBlink
GROUP_HANGTIME: 5
OPTIONS:
USE_ACL: True
SUB_ACL: DENY: 1
TGID_TS1_ACL: DENY: ALL
TGID_TS2_ACL: PERMIT: **9379**
ANNOUNCEMENT_LANGUAGE: fr_FR

Now let's open our second configuration file: **rules.py**
(rules of TG and Time Slots that you want to make available on your server)

nano /opt/FreeDMR/rules.py

```
BRIDGES = {

}
if __name__ == '__main__':
    from pprint import pprint
    pprint (BRIDGES)
```

and we will add here our routing rule (between the 2 {})

```
BRIDGES = {
            '9379': [ {'SYSTEM': 'THEGATE', 'TS': 2, 'TGID': 9379, 'ACTIVE': True, 'TIMEOUT': 10, 'TO_TYPE': 'NONE', 'ON': [9379], ' OFF ': [],' RESET ': []},
                    ],
            }
if __name__ == '__main__':
    from pprint import pprint
    pprint (BRIDGES)
```

CAUTION: As soon as you activate a PEER zone in **hblink.cfg**, you must have configured at least one routing line corresponding to this PEER connection in **rules.py**.

[**THEGATE**]
MODE: PEER
ENABLED: **True**

You must at least have a line like:
'9379': [{'SYSTEM': 'THEGATE', 'TS': 2, 'TGID': 9379, 'ACTIVE': True, 'TIMEOUT': 10, 'TO_TYPE': 'NONE', 'ON': [9379], ' OFF ': [],' RESET ': []},

A brace, a square bracket, a missing comma and your server will not launch!

Your server is now configured to collect 1TG in PEER mode from THEGATE and make it available on your own server.

The next step is to automate the startup of the FreeDMR server as a "service" by creating the file **freedmr.service** situated in **/lib / systemd / system / freedmr.service**

To do this, we open an empty file with the nano command

nano /lib/systemd/system/freedmr.service

Then we copy the content below inside

[Unit]
Description = FreeDMR Server After =
network-online.target syslog.target Wants =
network-online.target

[Service]
StandardOutput = null
WorkingDirectory = / opt / FreeDMR
RestartSec = 3
ExecStart = / usr / bin / python3 /opt/FreeDMR/bridge_master.py
Restart = on-abort

[Install]
WantedBy = multi-user.target

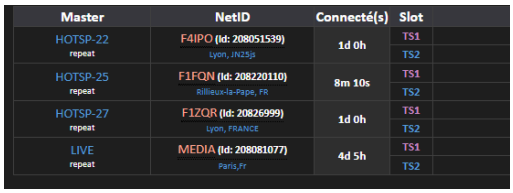
Once this file has been saved, you must: activate it, start it and finally test the start of the server with the following commands:

systemctl enable freedmr
systemctl start freedmr
systemctl status freedmr

As much the first 2 commands, when you run them, are not verbose as the 3rd takes care of checking the correct syntax of your hblink.cfg and rules.py files. A green dot tells you that everything is OK.
A red point and you will have to look for the problem in your configuration files and this can be limited to forgetting a single comma or a check mark !!!

```
root@dmrgatelink:/opt/FreeDMR# systemctl status freedmr.service
● freedmr.service - FreeDMR Server
   Loaded: loaded (/lib/systemd/system/freedmr.service; enabled; vendor preset:
   Active: active (running) since Tue 2021-10-12 15:51:14 CEST; 1 day 19h ago
 Main PID: 625 (/opt/FreeDMR/br)
    Tasks: 2 (limit: 2319)
   Memory: 255.8M
   CGroup: /system.slice/freedmr.service
           └─625 /opt/FreeDMR/bridge_master.py
```

At this point you should be logged into TheGate server and you can see that your server is logged in as a simple hotspot.
For the example, my test server is here represented by the line F1FQN on Hotspot-27



4-Creation of the reception area for guest hotspots

By the way, speaking of hotspots, you certainly want to be able to host guest connections on your server that will use your local TG or the TGs that you have imported from other servers (this is also the purpose of the manipulation)

This is where there is an important modification compared to the 1st generation HBlink server.
In the old versions, if you wanted to distribute a set of TGs available on your server, you had to manually create as many hotspot zones (each with a different communication port) as “subscribers”, otherwise a connection on the default zone (port 62031) only gave you access to one or 2 TGs fixed once and for all.

FreeDMR meanwhile, allows dynamic management of connected hotspots. To put it simply, all the hotspots connect to port 62031 and by a “reverse proxy” system, are placed in a personalized zone in which you can freely choose to use 2 TGs among those offered on your server. This mechanism is set up thanks to the file**/opt / FreeDMR / hotspot_proxy_v2.py**

Inside this file we find this:

```
#*** CONFIG HERE ***

    Master = "127.0.0.1"
    ListenPort = 62031
    # '' = all IPv4, '::' = all IPv4 and IPv6 (Dual Stack)
    ListenIP = ''
    DestportStart = 54000
    DestPortEnd = 54100
    Timeout = 30
    Stats = False
    Debug = False
    ClientInfo = False
    BlackList = [1234567]

```

In the file **/opt / FreeDMR / hblink.cfg** on the last line of the section **[SYSTEM],** you can adjust the number of hotspots you want to host**. By default this number is 100 (GENERATOR: 100).**
Of course, you can, according to your choice, reduce the number of ports in reservation. For our 100 connections, it will therefore create and reserve 100 connection ports from 54000 to 54100.

(If you want to host only 10 hotspots, only open from 54000 to 54010)

1er hotspot which will connect to your server on the default port **62031** will therefore be dynamically and randomly redirected to a free port between 54000 and 54100, the same for the second and the following ones ...

Note that these ports will now be reserved and should never be used for any connection to your server.

We are now going to create a service so that the reverse-proxy is launched when the server starts.
To do this, we open an empty file with the nano command:

nano /lib/systemd/system/hotspot_proxy.service

Then we copy the content below inside

[Unit]
Description = FreeDMR Hotspot_proxy After =
network-online.target syslog.target Wants =
network-online.target

[Service]
StandardOutput = null
WorkingDirectory = / opt / FreeDMR
RestartSec = 3
ExecStart = / usr / bin / python3 /opt/FreeDMR/hotspot_proxy_v2.py
Restart = on-abort

[Install]
WantedBy = multi-user.target

Once this file has been saved, you must: activate it, start it and finally test the startup of the hotspot_proxy with the following commands:

systemctl enable hotspot_proxy
systemctl start hotspot_proxy
systemctl status hotspot_proxy

```
root@dmrgatelink:/opt/FreeDMR# systemctl status hotspot_proxy.service
● hotspot_proxy.service - FreeDMR Hotspot_proxy
   Loaded: loaded (/lib/systemd/system/hotspot_proxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-10-12 15:51:14 CEST; 1 day 20h ago
 Main PID: 635 (/opt/FreeDMR/ho)
    Tasks: 1 (limit: 2319)
   Memory: 18.4M
   CGroup: /system.slice/hotspot_proxy.service
           └─635 /opt/FreeDMR/hotspot_proxy_v2.py
```

5-Creation of a zone dedicated to a hotspot on a fixed IP port

We have just seen that the server can dynamically host a certain number of guest connections (100 by default) whose incoming connection is invariably on the port **62031** and whose return port is random between **54000 and 54100 (for 100 hotspots)**

It could also be interesting (in addition to these dynamic areas) to be able to make one or more fixed areas available to connect specific hotspots to work. on a dedicated fixed port
.

In this case, you can create a new static SYSTEM zone that we will rename and to which we will assign a fixed port (here example of the port **55555**)

[**NAME OF YOUR DEDICATED ZONE**]
MODE: MASTER
ENABLED: True
REPEAT: True
MAX_PEERS: 1
EXPORT_AMBE: False
IP:
HARBOR: **55555** **(must not be in the range 54000 to 54100)**
PASSPHRASE: **a private password or the traditional passw0rd**
GROUP_HANGTIME: 5
USE_ACL: True
REG_ACL: DENY: 1
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
DEFAULT_UA_TIMER: 60
SINGLE_MODE: True
VOICE_IDENT: True
TS1_STATIC:
TS2_STATIC:
DEFAULT_REFLECTOR: 0
ANNOUNCEMENT_LANGUAGE: fr_FR
GENERATOR: 0 **The GENERATOR must remain fixed at 0**

Here is your server is now ready to accommodate a hundred connections and at this point, is fully operational.

6-Creating your server monitor - Dashboard

You are missing something important to know at a glance the activity of your server, namely the monitor also called DASHBOARD, in short the dashboard in Web format of your connections and that's what we will install now.

It is important to know that the monitor is not developed by the FreeDMR team but by an independent OM (SP2ONG), indeed the server and the monitor are two separate things.

Some SYSOP personalize their Monitor (Dashboard), for example that of the "TheGate" server that you can visit at this address
https://thegate.hblink.net:8080

By default after installation and without monitor customization, your dashboard will look like this:

We will now discuss the installation of the monitor. For this we will create a local installation directory on our server:

cd / opt
git clone https://github.com/sp2ong/HBMonv2

This command imports the installation directory locally on your server. Following the launch of this command, a directory**HBMonV2** will be created under **/opt**

Then we start the installation:

cd / opt / HBMonv2
chmod + x install.sh
. /install.sh
cp config_SAMPLE.py config.py

We will have to install now **PHP** and one **Web server**, which can be as desired **lighttpd**, **ngnix**, Where **apache2** !

Installation of php:
apt install php

Installation of the apache2 server:
apt install apache2

The installation of the WEB server had to create the directory **/var / www / html**

Let's go in to clean it up

cd / var / www / html
rm *. *

The html directory is now empty

We return to / opt / HBMonv2:

cd / opt / HBMonv2
cp -R html / var / www /

With this command, we recursively copy the content of **/opt / HBMonv2 / html** in **/ var / www / html**

You can now test your dashboard by connecting with the IP address or DNS of your server: **http: // Ip_de_your_serveur**

- In the directory **/var / www / html / include /** the file must be there **config.php**

By editing this file, you can change the colors and give a name to your dashboard

- In the directory **/var / www / html** the file must be there **buttons.html** to personalize your menus.

- In the directory **/var / www / html / img /** the file must be there **logo.png** that you can replace.

Finally, another file whose size must be checked and cleaned is lastheard.log (this is the file of the last OM's connected more than 2 seconds on the server). This task is performed automatically and daily according to the parameters included in the file **lastheard.**

cp / opt / HBMonv2 / utils / lastheard /etc/cron.daily/
chmod + x /etc/cron.daily/lastheard

As for all the other services, the monitor is no exception to the rule and it will be necessary to create a "service" so that it starts automatically.

The file **hbmon.service** is already configured, you just need to copy it to the services directory.

cp /opt/HBMonv2/utils/hbmon.service / lib / systemd / system /

Now you have to: activate it, start it and finally test the start of the hbmon monitor with the following commands:

systemctl enable hbmon
systemctl start hbmon
systemctl status hbmon

```
root@dmrgatelink:/opt/HBMonv2# systemctl status hbmon
● hbmon.service - HBMonitor
   Loaded: loaded (/lib/systemd/system/hbmon.service; enabled; vendor preset: en
   Active: active (running) since Tue 2021-10-12 15:51:14 CEST; 1 day 22h ago
 Main PID: 631 (python3)
    Tasks: 1 (limit: 2319)
   Memory: 310.7M
   CGroup: /system.slice/hbmon.service
           └─631 /usr/bin/python3 /opt/HBMonv2/monitor.py
```

7-Password protection of your Dashboard

If your server is private, or you just don't want curious people watching what's going on on your server, you might want to protect it with a username and password. In this case

You must create an empty file **.htpasswd** in / etc / apache2 with the command **touch** :
touch /etc/apache2/.htpasswd

Once this empty file has been created, it must be fed with an account and a password. Thatnot done with a text editor but with the command **htpasswd** (without the point this time).

htpasswd /etc/apache2/.htpasswd me (me or other being the name of your user)

The command then asks you to enter a password and confirm it. You can verify that your file **.htpasswd** has been written with the user "me" followed by his encrypted password.

cat /etc/apache2/.htpasswd

Now it is necessary to modify the configuration file of the Apache server which is apache2.conf

nano /etc/apache2/apache2.conf

For clarity, we will copy and paste the lines below at the end of the apache2.conf file and save

```
<Directory "/ var / www / html">
        Options Indexes FollowSymLinks
        AuthType Basic
        AuthName "Restricted access"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
</Directory>
```

Let's restart our Apache server with the usual command:

systemctl restart apache2

We check that everything is OK with

systemctl status apache2

The next time you log into your server, you will be asked for the username and password you have chosen.

8-Creation of an OPENBRIDGE with a partner server

We saw previously how to create a link **PEER** to a partner server. As a reminder, a link of this type behaves like the connection of a simple hotspot and at the server level,**it will be necessary to write a line in the rules.py file.** You will also need to write a line of options in your PISTAR (but we will come back)

OPENBRIDGE (OPB) is the best solution for connecting two partner servers. Indeed a dedicated channel is open between the 2 servers and you can potentially exchange your TGs without limitation. The OPENBRIDGE transmission protocol is optimized for this type of connection.
Apart from the technical aspects, an OPENBRIDGE connection is socially respectful because it requires you to contact the administrator of a partner server to exchange your IP addresses, your TGs, send and receive ports, password and especially your **Network ID**

An important aspect to remember is that with FreeDMR, an Openbridge connection, no line of "rules" has to be written (everything is automatic)

If you are a licensed radio amateur, you have 2 possibilities to set up a Freedmr server:

1- **For a public server**
   By being an integral part of this network and referenced as such. You will therefore need to request a unique network number (network ID) which will identify you as an official Freedmr server.
   This number will be essential for the Openbridge connection to work.

2- **For a private server other than FreeDMR**
   By wishing to benefit from the TG of the Freedmr network but without being officially affiliated and referenced.
   To do this, all you need to do is "invent" a network ID outside of those officially distributed by FreeDMR. Official “network IDs” start with 208 (for the country, 208 is France) +1 or 2 digits.
   If, for example, you invent a 10-digit ID like 4568214975, you can be almost sure that you will never come into conflict with another officially declared server.

Having said that, let's see how to establish an OPENBRIDGE (OPB) gateway with a partner server.

- Request a Network ID on Telegram by contacting Simon @hacknix
https://t.me/FreeDMR_Building_Server_Support

- Find a partner server offering the TG (s) that you want to use on your own server. If you want to create a new TG and make it available to the whole world, you will need to request it from Norman @NormanFreeDMR who will send you a form to fill out.

http://www.freedmr.uk/index.php/world-wide-talk-groups/

Ps: FreeDMR servers can potentially distribute all global TGs to you unless the server admin has decided (by filtering) to keep only those that are useful. It is however advisable (without your VPS at 5 € being too busy) to establish only 3 Openbridges at most (redundancy or to recover on another server, TGs that your first partner does not offer you).

Once your partner server has been found, contact its administrator and ask for an Openbridge link. I am not aware of a "directory" of FreeDMR server administrators but if you look around, you will find it!
It will send you an Openbridge section to install in your hblink.cfg or freedmr.cfg file (docker version). This file will look like this:

[OBP-PARTNER]
MODE: OPENBRIDGE
ENABLED: True
IP: PORT: 62100 (the IP: PORT and the TARGET_PORT can be identical or different) * The
NETWORK_ID: Network ID that your partner admin will communicate to you
PASSPHRASE: The password that your partner admin will give you The IP or
TARGET_IP: DNS of your partner server
TARGET_PORT: 62100 The IP port offered by your partner server USE_ACL:
True
SUB_ACL: DENY: 1
TGID_ACL: PERMIT: The list of TGs you want to retrieve; the others will be rejected **
RELAX_CHECKS: True
ENHANCED_OBP: True

*If you want to keep all the TGs from the partner server: TGID_ACL: PERMIT: ALL
* *Each time a new OPB is connected, it will be necessary to increase your **IP: PORT**

You must communicate your Network ID and the IP address of your server to the partner admin.

Important: Do not forget to register your **Network ID in GLOBAL section** from your hblink.cfg file **SERVER_ID: Your network ID**
Without it, you will be well connected but no stream will be transmitted.
Your file**hblink.cfg** will therefore evolve like this:

[GLOBAL]
PATH: ./
PING_TIME: 10
MAX_MISSED: 3
USE_ACL: True
REG_ACL: PERMIT: ALL
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
GEN_STAT_BRIDGES: True
ALLOW_NULL_PASSPHRASE: True
ANNOUNCEMENT_LANGUAGES: en_GB, en_US, es_ES, fr_FR, de_DE, dk_DK, it_IT, no_NO, pl_PL, se_SE, pt_PT, cy_GB, el_GR, CW
SERVER_ID: 0000 **- replace here with your network ID**

[REPORTS]
REPORT: True
REPORT_INTERVAL: 60
REPORT_PORT: 4321
REPORT_CLIENTS: 127.0.0.1

[LOGGER]
LOG_FILE: freedmr.log
LOG_HANDLERS: file-timed
LOG_LEVEL: INFO
LOG_NAME: FreeDMR

[ALIASES]
TRY_DOWNLOAD: True
PATH: ./
PEER_FILE: peer_ids.json
SUBSCRIBER_FILE: subscriber_ids.json
TGID_FILE: talkgroup_ids.json
PEER_URL: https://www.radioid.net/static/rptrs.json SUBSCRIBER_URL: https://www.radioid.net/static/users.json TGID_URL: http://downloads.freedmr.uk/downloads/talkgroup_ids.json STALE_DAYS: 7

########################
#   #
Mysql has been abandoned
#   #
########################

[MYSQL]
USE_MYSQL: False
USER: hblink
PASS: mypassword
DB: hblink
SERVER: 127.0.0.1
PORT: 3306
TABLE: repeaters

########################
#   #
OPENBRIDGE section
#   #
########################

[OBP-PARTNER]
MODE: OPENBRIDGE
ENABLED: True **If False your OPB is disabled**
IP: PORT: 62044 **(the IP: PORT and the TARGET_PORT can be identical or different) The**
NETWORK_ID: **Network ID that your partner admin will communicate to you**
PASSPHRASE: **The password that your partner admin will give you The IP or DNS**
TARGET_IP: **of your partner server**
TARGET_PORT: 62044 **The IP port offered by your partner server**
USE_ACL: True
SUB_ACL: DENY: 1
TGID_ACL: PERMIT: The **list of TGs you want to retrieve separated by commas**. The other TG will be rejected or**ALL** to keep all the TG
RELAX_CHECKS: True
ENHANCED_OBP: True

```
#######################################################
#                                                     #
# Hotspot zone generator #
#                                                     #
#######################################################

 [SYSTEM]
MODE: MASTER
ENABLED: True
REPEAT: True
MAX_PEERS: 1
EXPORT_AMBE: False
IP: 127.0.0.1
PORT: 54000
PASSPHRASE:
GROUP_HANGTIME: 5
USE_ACL: True
REG_ACL: DENY: 1
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
DEFAULT_UA_TIMER: 60
SINGLE_MODE: True
VOICE_IDENT: True
TS1_STATIC:
TS2_STATIC:
DEFAULT_REFLECTOR: 0
ANNOUNCEMENT_LANGUAGE: en_FR
GENERATOR: 100



########################
#                      #
# PEER section         #
#                      #
########################

[THEGATE]
MODE: PEER
ENABLED: True
LOOSE: False
EXPORT_AMBE: False
IP:
PORT: 54301
MASTER_IP: thegate.hblink.net
MASTER_PORT: 62031
PASSPHRASE: passw0rd
CALLSIGN: Your callsign
RADIO_ID: Your DMR Id CCS7 or CCS7 + suffix from 01 to 99
RX_FREQ: 449000000
TX_FREQ: 444000000
TX_POWER: 25
COLORCODE: 1
SLOTS: 2
LATITUDE: 46.000000 (to change)
LONGITUDE: 5.000000 (to change)
HEIGHT: 320
RENTAL: Your City, Country or QRA locator
DESCRIPTION: Virtual hotspot URL:
www.google.fr
SOFTWARE_ID: 20170620
PACKAGE_ID: MMDVM_HBlink
GROUP_HANGTIME: 5
OPTIONS:
USE_ACL: True
SUB_ACL: DENY: 1
TGID_TS1_ACL: DENY: ALL
TGID_TS2_ACL: PERMIT: 9379
ANNOUNCEMENT_LANGUAGE: fr_FR
```

9-Creation of an XLXPEER connection on an XLX gateway

There are servers that act as gateways between the different digital modes.
These are XLX servers.
C4FM Gateway <-> DMR, DSTAR <-> DMR

Example: The XLX933C gateway http://xlx933.dstar-france.fr/index.php?show=modules

Its IP is: 152.228.170.6 or its DNS: xlx933.dstar-france.fr
and in the table on the right you will find a column named DMR in which there are module names. These module names can be displayed as a 4-digit number or the name of the server followed by the letter of the alphabet corresponding to the last 2 digits of the module.

Example: xlx933-Y will be equivalent to xlx933 module 4025 25 because Y is the 25th letter of the alphabet.

A connection on the xlx933 module 4025 server will put your DMR server in touch with the YSF-France network which operates in C4FM.

Likewise, a connection to the xlx933 module 4003 server will connect your DMR server and the DSTAR-France network.

An XLXPEER connection is a variant of the PEER connection but has 5 particularities:

Fashion : **XLXPEER** instead of PEER
LOOSE: **True** while it is False for a PEER MASTER_PORT
connection: **62030**
SLOTS: **1**
You will need to create a TG number for this connection in the rules section

Recall :

SLOTS: **0** **TS1 only**
SLOTS: **1** **TS2 only**
SLOTS: **2** **TS1 or TS2**

Use TG9 / TS2 for writing the rule line

[**XLX933-DSTAR**]
FASHION: **XLXPEER**
ENABLED: True
LOOSE: **True**
EXPORT_AMBE: False
IP:
PORT: 54607 **An unused port on your server** MASTER_IP:
152.228.170.6 MASTER_PORT**: 62030**

PASSPHRASE: **passw0rd**
CALLSIGN: **Your callsign** RADIO_ID:
208xxxx **Your DMR ID CCS7** RX_FREQ:
44900000
TX_FREQ: 444000000
TX_POWER: 25
COLORCODE: 1
SLOTS: **1**
LATITUDE: **46.000000 (to change)**
LONGITUDE: **5.000000 (to be changed)** HEIGHT: 320
RENTAL: **Your city**
DESCRIPTION:
URL: http: //www.google; fr
SOFTWARE_ID: 20170620
PACKAGE_ID: MMDVM_FreeDMR
GROUP_HANGTIME: 5
4000 + the numerical position of the module in the alphabet - eg A = 4001
XLXMODULE: **4003**
USE_ACL: True
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
ANNOUNCEMENT_LANGUAGE: fr_FR


```
BRIDGES = {
                '9379': [{'SYSTEM': 'THEGATE', 'TS': 2, 'TGID': 9379, 'ACTIVE': True, 'TIMEOUT': 10, 'TO_TYPE': 'NONE', 'ON': [9379], 'OFF': [], 'RESET': []},],
               'TG No.': [{' SYSTEM ': 'XLX933-DSTAR ', 'TS': 2, 'TGID': 9, 'ACTIVE': True, 'TIMEOUT': 2, 'TO_TYPE': 'NONE', 'ON': [TG No.], 'OFF': [], 'RESET': []},],
                }
if __name__ == '__main__':
    from pprint import pprint
    pprint (BRIDGES)
```

10-Creation of one or more TGs on your server

You would surely also like to be able to create your or your own TG.

Nothing could be simpler As indicated above, all you have to do is fill in the following fields in the [SYSTEM] area:

```
#################################################
#                                               #
# Hotspot zone generator #
#                                               #
#################################################
```

[SYSTEM]
MODE: MASTER
ENABLED: True
REPEAT: True
MAX_PEERS: 1
EXPORT_AMBE: False
IP: 127.0.0.1
PORT: 54000
PASSPHRASE:
GROUP_HANGTIME: 5
USE_ACL: True
REG_ACL: DENY: 1
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
DEFAULT_UA_TIMER: 60
SINGLE_MODE: True
VOICE_IDENT: True
TS1_STATIC: **Here the TG (s) you want to create and make available for TS1** TS2_STATIC: **Here the TG (s) you want to create and make available for TS2** DEFAULT_REFLECTOR: 0

ANNOUNCEMENT_LANGUAGE**: en_FR**
GENERATOR: 100

The TGs to be created must be separated by commas

Be careful unlike HBLINK in the rules.py file:

- The **TG field** (example of TG 9379) should only be **digital** ! (no label TG9379)

- Each PEER or XLXPEER must correspond to a new line of rules
- No line of "rules" is to be written for OPB connections

11- Installation of a parrot

No, not that one!

The parrot (Parrot) is a service that allows you to send you the echo of your modulation on a particular TG (TG9990).

To build this we need:

- Install a zone **PEER** that we will name [**ECHO**] in our file **hblink.cfg**
- Create a file **playback.cfg** to which our PEER [ECHO] will be connected
- To create a **parrot service** for the parrot to launch automatically with the server.

Normally during your FreeDMR installation, the script **install.sh** has already installed all the software dependencies necessary for the correct functioning of your parrot.

However and if necessary, these dependencies are in the requirements.txt file **/ opt/FreeDMR/requirements.txt** and can, if necessary, be installed.

Now let's create the zone **playback.cfg** :

nano /opt/FreeDMR/playback.cfg with the following content:

[GLOBAL]
PATH: ./
PING_TIME: 10
MAX_MISSED: 5
USE_ACL: True
REG_ACL: PERMIT: ALL
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL TGID_TS2_ACL: PERMIT: ALL GEN_STAT_BRIDGES: False
ALLOW_NULL_PASSPHRASE: False ANNOUNCEMENT_LANGUAGES: en_GB, en_US, es_ES, fr_FR, de_DE, dk_DK, it_SEIT, pl_GB, it_SEIT, pl_GB, it_SEIT, pl_GB, 000_SEIT, pl_GB_DK, fr_FR, de_DE, dk_DK, it_0000, 0000, pl_PT_GB, p_t_, 000

[REPORTS]
REPORT: False
REPORT_INTERVAL: 60
REPORT_PORT: 4322
REPORT_CLIENTS: 127.0.0.1

[LOGGER]
LOG_FILE: /var/log/parrot.log
LOG_HANDLERS: file-timed
LOG_LEVEL: INFO
LOG_NAME: Parrot

[ALIASES]
TRY_DOWNLOAD: False
PATH: ./
PEER_FILE: peer_ids.json
SUBSCRIBER_FILE: subscriber_ids.json
TGID_FILE: talkgroup_ids.json
TGID_URL: http://downloads.freedmr.uk/downloads/talkgroup_ids.json
PEER_URL: https://database.radioid.net/static/rptrs.json
SUBSCRIBER_URL: https://database.radioid.net/static/users.json
STALE_DAYS: 1

[PARROT]
MODE: MASTER
ENABLED: True
REPEAT: True
MAX_PEERS: 1
EXPORT_AMBE: False
IP: **127.0.0.1**
HARBOR: **54915**
PASSPHRASE: passw0rd
GROUP_HANGTIME: 5
USE_ACL: True
REG_ACL: DENY: 1
SUB_ACL: DENY: 1
TGID_TS1_ACL: DENY: ALL
TGID_TS2_ACL: PERMIT:**9990**
DEFAULT_UA_TIMER: 10
SINGLE_MODE: True
VOICE_IDENT: False

Then edit the area **ECHO** from our file **hblink.cfg**

[**ECHO**]
MODE: PEER
ENABLED: True
LOOSE: False
EXPORT_AMBE: False
IP:
HARBOR: **54916**
MASTER_IP: **127.0.0.1**
MASTER_PORT: **54915** **(here the port must be identical to that of the PARROT zone of the playback.cfg file)**
PASSPHRASE: passw0rd
CALLSIGN: **ECHO**
RADIO_ID: **9990** **(Imperative)**
RX_FREQ: 430500000
TX_FREQ: 439900000
TX_POWER: 25
COLORCODE: 1
SLOTS: **1 (Reminder 0 =TS1 only - 1 = TS2 only - 2 = TS1 or TS2)** LATITUDE: 45.000000

LONGITUDE: 004.00000
HEIGHT: 320
LOCATION: LYON (69)
DESCRIPTION: ECHO
Url:
SOFTWARE_ID: 20170620
PACKAGE_ID: MMDVM_FreeDMR
GROUP_HANGTIME: 5
OPTIONS:
USE_ACL: True
SUB_ACL: DENY: 1
TGID_TS1_ACL: PERMIT: ALL
TGID_TS2_ACL: PERMIT: ALL
ANNOUNCEMENT_LANGUAGE: fr_FR

Let's add the following line to our file **rules.py**

```
'9990': [
        {'SYSTEM': 'ECHO', 'TS': 2, 'TGID': 9990, 'ACTIVE': True, 'TIMEOUT': 2, 'TO_TYPE': 'NONE', 'ON': [9990], ' OFF ': [],' RESET ': []},],
```

Now let's create the service in order to launch the "Parrot" automatically:

Let's edit the file **parrot.service:**

nano /lib/systemd/system/parrot.service and insert the following lines:

```
[Unit]
Description = HB bridge all Service After =
network-online.target syslog.target Wants =
network-online.target

[Service]
StandardOutput = null
WorkingDirectory = / opt / FreeDMR
RestartSec = 3
ExecStart = / usr / bin / python3 /opt/FreeDMR/playback.py -c /opt/FreeDMR/playback.cfg
Restart = on-abort
[Install]

WantedBy = multi-user.target
```

Starting the PARROT service:

Activate the PARROT service:
systemctl enable parrot

Start the PARROT service:
systemctl start parrot

Test the functioning of the PARROT service (which should not return errors)
systemctl status parrot

Do not forget to restart your FreeDMR server in order to take into account the modification of your files **hblink.cfr and rules.py**

systemctl restart freedmr
systemctl status freedmr

After restarting, if your ECHO zone is green on your Dashboard, position your TX on the **TG9900 / TS2**, speak up and listen to your feedback!