How to Create a Multi-Mode XReflector

By Dennis Mojado, AD6DM

ad6dm@arrl.net

February 2019

Last updated: May 10, 2019

Table of Contents

Introduction

Documentation on this subject is spread throughout the Internet, with several good sources, but very few end-to-end sources.

What do we want to do?

We would like to set up an Internet-based multi-mode reflector that allows a ham radio operator with a DStar, Fusion, or DMR radio to enter a reflector, room, or talkgroup, and be heard by all the others in the same “net”, regardless of the others’ digital voice mode.

So what we are building is called an XLX Multi-Mode Reflector.

Out of the box (so to speak) this XLX reflector is capable of accepting DStar and Fusion audio streams. DMR is also possible, but this is not necessarily Brandmeister DMR. You will need something extra (YSF2DMR) to connect to regular BM DMR talkgroups (more on that later).

Kings of Digital, San Antonio Digital Radio, and various other groups have made good PDFs and website pages with lots of great information on how to get this going. But they gloss over some critical path details that require extensive online searching to understand. This is understandable, because once it works it seems intuitive that it should have worked that way. But for the learner, this is a difficult path that often resorts to cries for help in forum discussion boards, looking at example configuration files, and even reverse-engineering C++ code to understand what parameters mean and hopefully gleaning something usable to plug into a configuration file.

Documenting this all is a daunting task, but I will share some insights in this How-To in the hopes that I will be able to contribute to the collective knowledge on how to get more hams on digital voice radio.

This will not be a quick read, nor will it be an exacting step-by-step how-to for all different scenarios. However, it will contain some information that I gathered from several days of scouring for answers to get my group of hams talking to each other.

Debian 9 Server

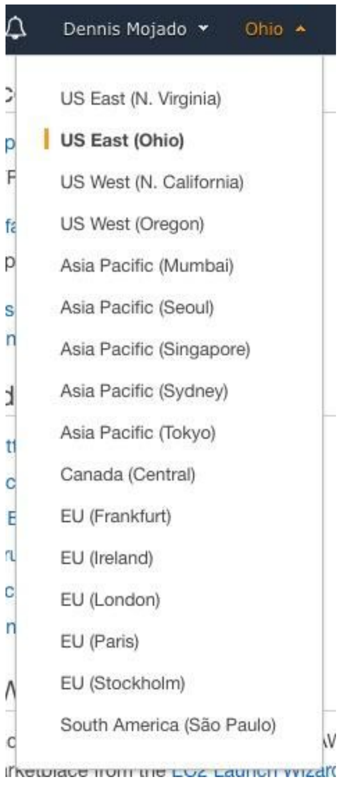
First and foremost, a person needs to set up a Debian 9 (Stretch) server.

Debian can be downloaded from https://www.debian.org/distrib/netinst . Most likely you will want the amd64 build for most modern computers.

I decided to do this on Amazon Web Services Elastic Compute Cloud (AWS EC2) because I didn't want to have a reflector at the mercy of my unreliable cable Internet nor at the mercy of my power company or consumer-grade home network infrastructure. YMMV, if you want to set this up at home with your own Debian server, that allows for greater control. But for XLX367, I would rather outsource the uptime service level and geographic availability to the cloud where large companies specialize in this. If you don't go the AWS route, you can skip the next few sections.

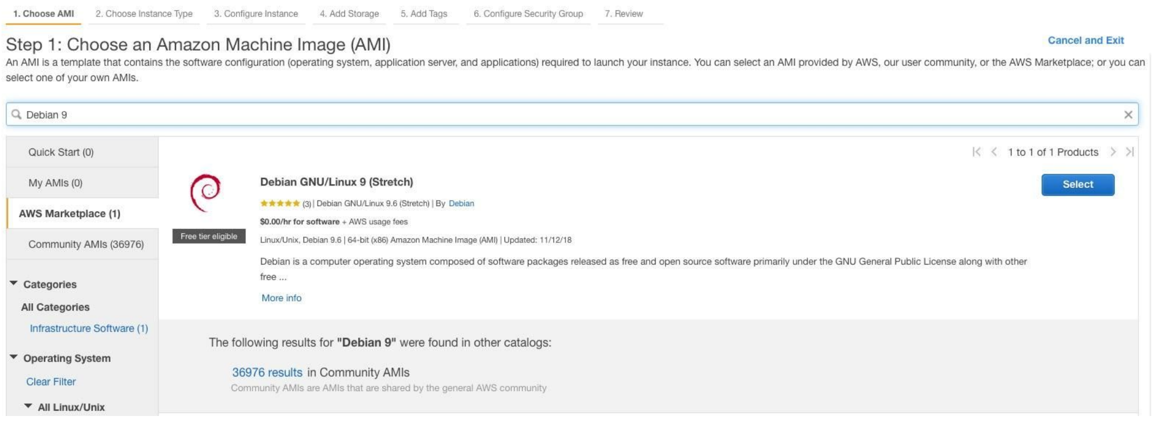
AWS Setup

If you don't know what AWS is, you're going to have to pause here and learn how to get an AWS account, and learn a little bit about how to manage that account. That How-To is outside the scope of this document.



First I went to AWS Console and selected a region central to all my users. I picked Ohio (US-East 2) because the target users are scattered throughout the country and I felt a central USA location (not really all that "central") would be the best for least network latency.

I chose "Launch New Instance" and went to the AWS Marketplace to find the Debian 9 AMI. It's a base image of Debian version 9.6, was released on November 10th, 2018.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Debian 9

Debian GNU/Linux 9 (Stretch)

★★★★★ (3) | Debian GNU/Linux 9.6 (Stretch) | By Debian

$0.00/hr for software + AWS usage fees

Linux/Unix, Debian 9.6 | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 11/12/18

Debian is a computer operating system composed of software packages released as free and open source software primarily under the GNU General Public License along with other free ...

The following results for **"Debian 9"** were found in other catalogs:

36976 results in Community AMIs

Community AMIs are AMIs that are shared by the general AWS community

I then chose a t2.micro instance. This instance is "eligible for free tier" but I've long since lost that introductory privilege. That means I'll be spending about $10/month for this server instance.

Perform the usual steps of setting up an instance, such as assigning an IAM role, and downloading the keys so you can later ssh to the instance.

Once that's fired up (several minutes), you can begin configuring the network.

Use an Elastic IP on AWS

Optional but very handy:

If you want to easily swap out instances but keep the same IP address, set up an Elastic IP right after you create the new instance, and assign that public IP to the new Debian server. This is good if you want test an upgrade or do a total rebuild in a different instance, but don't want to wait days for a DNS IP change, or weeks for a registration IP change.

Set Up Your FQDN on Your Own Registered Domain

After setting up your instance with fresh Debian 9, you should have a fully qualified domain name (FQDN) in mind. In XLX367 JerryNet's case, I decided on *xrf367.ad6dm.net*.

Go to your DNS management page, typically in your domain registrar, and set up the DNS record for your new Elastic IP address. It takes the form of

```
xrf367     A     18.216.66.72
ysf        A     18.216.66.72
```

(I put the YSF Reflector and XLX reflector on the same server.)

You will need both the domain and the IP to register with **XReflector** and **YSF Reflector** online directories.

Firewalls (Security Group) Settings

AWS instances, and most other cloud setups (e.g. Digital Ocean, Vultr, etc) don't open new server instances to the world on all ports. So you have to set up firewall rules (a.k.a. Security Groups in AWS terminology) to allow good traffic in.

Even on Debian installs at your home physical box, you will need to configure your home network to allow certain types of traffic to come in through the router to your Debian linux machine. Here are the firewall allow rules that you need to configure. In my case, this was in AWS' Security Group Settings

- TCP port 80 (http)
- TCP port 443 (https)
- TCP port 22 (ssh)
- UDP port 10001 (json interface XLX Core)

- UDP port 10002 (XLX interlink)
- UDP port 10100 (AMBE controller port)
- UDP port 10101 - 10199 (AMBE transcoding port)
- UDP port 30001 (DExtra protocol)
- UDP port 20001 (DPlus protocol)
- UDP port 30051 (DCS protocol)
- UDP port 62030 (MMDVM protocol)
- UDP port 8880 (DMR+ DMO mode)
- UDP port 42000 (YSF Reflector)

Note: It's also helpful if you aren't restricting SSH to certain IPs, to install *fail2ban*, which will ban IP addresses that are attempting to login to your server. Default settings for fail2ban are ban an IP for 10 minutes if it fails login 5 times over a 10 minute period. Configuration of fail2ban is not critical to this document, so I'll leave that for you to search on the Googs.

Logging into your new Debian Server

When you first set up a new AWS instance, it prompted you to download a PEM key. You can login to your new server by doing:

```
ssh -i /path/to/YourKey.pem admin@yourserver.ip.address
```

Optional: SSL Secure your Dashboards

Like most admins, I hate unencrypted websites. Everything should be using SSL, especially with the ease of getting a SSL certificate through LetsEncrypt.

```
sudo apt-get install python-certbot-nginx
```

I prefer managing SSL through an SSL terminator, rather than having everything go through Apache. In this case, I use nginx as the termination server.

```
sudo apt-get install nginx
```

After installation, you will need to configure nginx to point to your letsencrypt certificates. Our config is here:

https://gist.github.com/denmojo/8cdab21800fd25012c58a8b2fa779052

In order to get the LetsEncrypt certificate, I ran:

```
sudo certbot --authenticator standalone --installer nginx -d
xrf367.ad6dm.net -d ysf.ad6dm.net --pre-hook "service nginx stop"
--post-hook "service nginx start"
```

You will also have to change the default Apache ports to listen on 8080 after you have run the scripts below. (Of course, you can change these in the nginx config and apache config to any other unused port you want.)

These are found in:

```
/etc/apache2/sites-enabled/yourdomainname.conf
```

and

```
/etc/apache2/ports.conf
```

Modify the ports in those files to listen on 8080.

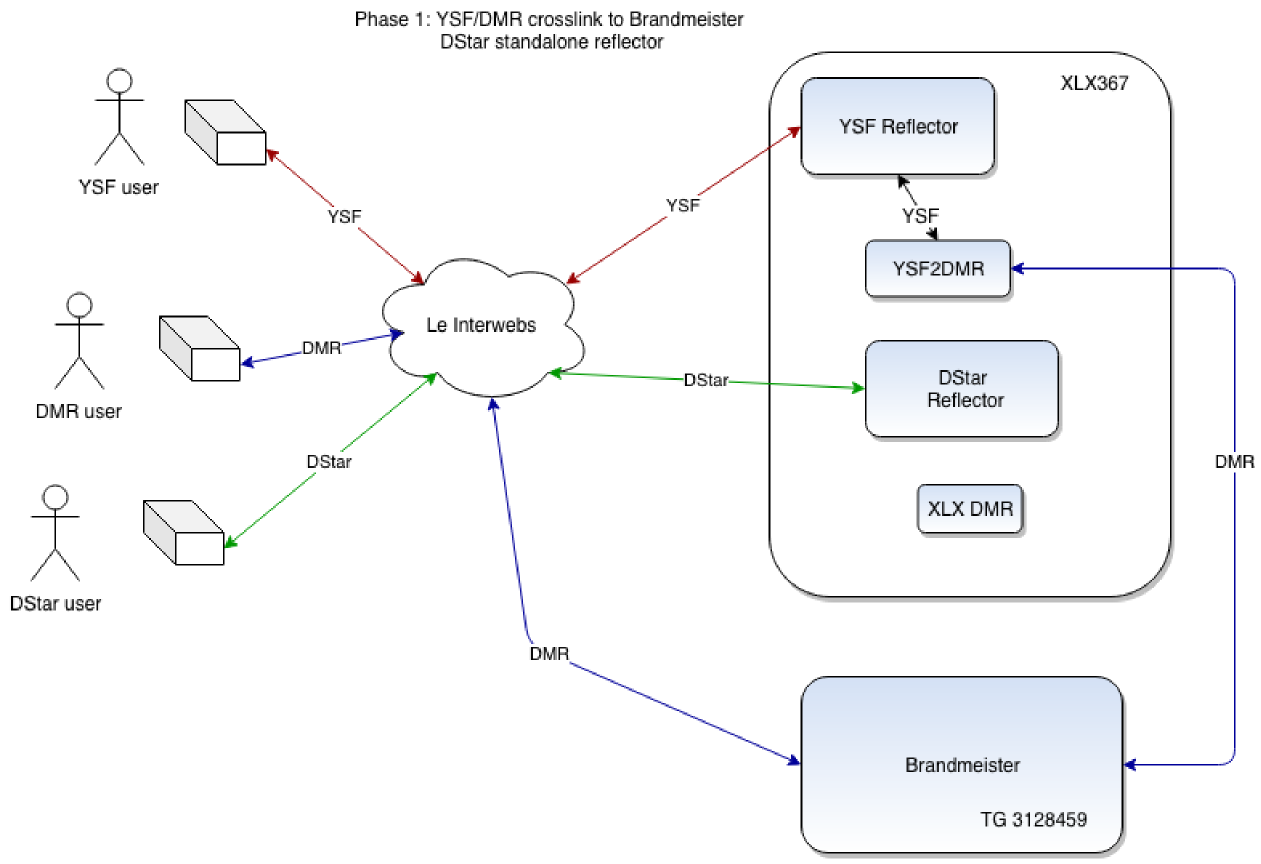
Phase 1: XLX DStar Reflector and YSF (to DMR) Reflector

That was some tedious stuff, huh? It was also a test of your resolve. If you got this far, you may yet get an XLX Reflector running! Now that you have a running Debian 9 server, you can use a convenience script by Ben N5AMD to install the XLX Reflector software and dashboard, as well as YSF Reflector and YSF2DMR.

As you wait for approvals and registrations, it is best to approach this project in phases. This is phase 1, where you will end up with a YSF to DMR crosslink, and a separate DStar standalone reflector.

Diagram of Phase 1

Here is a diagram of what this phase will look like.



(The example talkgroup TG 3128459 above is AD6DM's secondary DMR ID. During the Talkgroup Request phase, Brandmeister support suggested our group use this for testing and for their admins to measure traffic to justify approval. **DO NOT LINK TO THIS TALKGROUP IN YOUR SERVER CONFIG.** Obtain your own talkgroup number.)

Install the Multi-Reflector Server Applications

To run a multi-mode server including DStar, YSF, and YSF2DMR, run the following script found at

https://github.com/n5amd/Multi-Reflector-Installer

```
git clone https://github.com/n5amd/Multi-Reflector-Installer

cd Multi-Reflector-Installer

./Multi-Reflector-Installer.sh
```

Follow the prompts, and enter initial setup info. It will install and fill the initial information, but will require configuration after it's done.

Configuring your XLX Reflector

Update your configuration in `/var/www/xlxd/pgs/config.inc.php` in the parts marked in yellow:

```
<?php
/*
Possible values for IPModus

HideIP
ShowFullIP
ShowLast1ByteOfIP
ShowLast2ByteOfIP
ShowLast3ByteOfIP

*/

$Service    = array();
$CallingHome = array();
$PageOptions = array();
$VNStat     = array();

$PageOptions['ContactEmail']                          = 'ad6dm@arrl.net';
// Support E-Mail address

$PageOptions['DashboardVersion']                      = '2.4.0';       //
Dashboard Version

$PageOptions['PageRefreshActive']                     = true;     //
Activate automatic refresh
$PageOptions['PageRefreshDelay']                      = '10000';       //
Page refresh time in miliseconds

$PageOptions['RepeatersPage'] = array();
$PageOptions['RepeatersPage']['LimitTo']              = 99;       // Number
of Repeaters to show
```

```
$PageOptions['RepeatersPage']['IPModus']         = 
'ShowLast1ByteOfIP';    // See possible options above
$PageOptions['RepeatersPage']['MasqueradeCharacter'] = '*';       // 
Character used for  masquerade

$PageOptions['PeerPage'] = array();
$PageOptions['PeerPage']['LimitTo']              = 99;       // Number 
of peers to show
$PageOptions['PeerPage']['IPModus']              = 
'ShowLast1ByteOfIP';    // See possible options above
$PageOptions['PeerPage']['MasqueradeCharacter']  = '*';      // 
Character used for  masquerade

$PageOptions['LastHeardPage']['LimitTo']         = 39;       // Number 
of stations to show

$PageOptions['ModuleNames'] = array();                       // Module 
nomination
$PageOptions['ModuleNames']['A']                 = 'Int.';
$PageOptions['ModuleNames']['B']                 = 'Regional';
$PageOptions['ModuleNames']['C']                 = 'National';
$PageOptions['ModuleNames']['D']                 = '';

$PageOptions['MetaDescription']                  = 'XLX is a D-Star 
Reflector System for Ham Radio Operators.';    // Meta Tag Values, 
usefull for Search Engine
$PageOptions['MetaKeywords']                     = 'Ham Radio, D-Star, 
XReflector, XLX, XRF, DCS, REF, ';     // Meta Tag Values, usefull 
forSearch Engine
$PageOptions['MetaAuthor']                       = 'LX1IQ';
                            // Meta Tag Values, usefull for Search 
Engine
$PageOptions['MetaRevisit']                      = 'After 30 Days';
                            // Meta Tag Values, usefull for Search 
Engine
$PageOptions['MetaRobots']                       = 'index,follow';
                            // Meta Tag Values, usefull for Search 
Engine

$PageOptions['UserPage']['ShowFilter']           = true;
                       // Show Filter on Users page
```

```
$PageOptions['Traffic']['Show']                        = false;
                            // Enable vnstat traffic statistics

$PageOptions['CustomTXT']                              = 'JerryNet XLX
Reflector';                      // custom text in your header

$Service['PIDFile']                                    =
'/var/log/xlxd.pid';
$Service['XMLFile']                                    =
'/var/log/xlxd.xml';

$CallingHome['Active']                                 = false;
           // xlx phone home, true or false
$CallingHome['MyDashBoardURL']                         =
'https://xrf367.ad6dm.net';           // dashboard url
$CallingHome['ServerURL']                              =
'http://xlxapi.rlx.lu/api.php';        // database server, do not
change !!!!
$CallingHome['PushDelay']                              = 600;
      // push delay in seconds
$CallingHome['Country']                                = "USA";
      // Country
$CallingHome['Comment']                                = "JerryNet Reflector
on XRF367";                // Comment. Max 100 character
$CallingHome['HashFile']                               =
"/tmp/callinghome.php";          // Make sure the apache user has read
and write permissions in this folder.
$CallingHome['LastCallHomefile']                       =
"/tmp/lastcallhome.php";         // lastcallhome.php can remain in the
tmp folder
$CallingHome['OverrideIPAddress']                      = "";
      // Insert your IP address here. Leave blank for autodetection.
No need to enter a fake address.
$CallingHome['InterlinkFile']                          =
"/xlxd/xlxd.interlink";          // Path to interlink file

$VNStat['Interfaces']                                  = array();
$VNStat['Interfaces'][0]['Name']                       = 'eth0';
$VNStat['Interfaces'][0]['Address']                    = 'eth0';
$VNStat['Binary']                                      = '/usr/bin/vnstat';
```

```
/*
include an extra config file for people who dont like to mess with
shipped config.ing.php
this makes updating dashboard from git a little bit easier
*/

if (file_exists("../config.inc.php")) {
 include ("../config.inc.php");
}

?>
```

Do not turn "`Active`" to `true` until you have double-checked your desired reflector number to avoid duplication of an existing reflector number, and ready to go live. (See below for activation section.)

Testing Your XLX Reflector for DStar

DStar will work with your new XLX Reflector out of the box.

It will be able to service "multi-mode" DStar connections. That means, it will accept **DPlus (REF) connections, DCS connections, and DExtra (XRF) connections**. However, it won't be found by other hams from repeaters or their hotspots because it was not officially active or added to their hosts files yet.

However, you can test DStar on your own hotspot by "hacking" the hosts file and including the IP address of your XLX Reflector in that file under the number you choose. It needs to be a free XLX number, and most of them are taken already.

ssh to your hotspot as user "pi-star":

```
$ sudo su -
```

```
# rpi-rw
```

```
# vi /root/DExtra_Hosts.txt
```

This will enable you to override the known entry and test your XLX reflector before it is added to the registry.

Requesting Add to the XLX Directory (deprecated)

Update: On February 27, 2019, K6KD stopped listing XLX-based XRF reflectors, and now lists only non-XLX "traditional" XRF reflectors for double-checking against intended XRF number usage. See his comments here and here.

The XLX development team provides an automated way to list active/known XLX reflectors, and this can be found on any XLX dashboard's "Reflector List". Check out our reflector list: https://xrf367.ad6dm.net/index.php?show=reflectors .

Choose an unused/unlisted number when creating your own XLX reflector to avoid duplication and confusion. There is no longer any "registration" request process for XLX. Simply choose your XRF number, double-check that the number isn't taken by a non-XLX "traditional" reflector at http://xrefl.net/, and if it is free, *activate* your XLX reflector (set *Active* config parameter mentioned above to `true`). It will notify the rest, and will become part of their lists.

Of course, you can try to use a number that appears available in the XLX reflector list, but happens to be already reserved in xrefl.net. If you do this, you will not be able to use the XRF protocol for that particular number. You will only get to use REF and DCS protocols for that number.

I must admit, my initial understanding of this reflector list concept was incorrect: I thought that by being added to the xrefl directory, somehow that was referenced as the master list by other directories and XLX sites, and I thought only after addition were you permitted to "activate" your reflector. This is not the way it works, and you only need to reference a dashboard reflector list of any up XLX server. You don't do this to gain permission to activate, but to avoid number duplications which would make your new XLX reflector unreachable by the public.

Original (deprecated) request process was as follows.

~~This is a step that requires patience because it requires review and approval by the XLX directory maintainers. Go to the "Reflector List" of any XLX site (for example https://xrf367.ad6dm.net) and look for any unused/unlisted reflector number. Cross-reference this number against Kings of Digital xreflector directory at http://www.xrefl.net . If the reflector number you have in mind appears to be unused on those two directories, you can try to apply for that reflector number.~~

~~You must go to:~~

~~http://xrefl.boards.net/thread/2/request-adding-changing-directory-xrefl~~

~~And add a reply to the end of the thread asking for your XLX reflector to be added to the core host files, and registered as an XLX reflector. Only then after this is done by the XReflector Directory maintainers can you activate your XLX reflector.~~

~~The request on the board goes in the format:~~

```
~~1) Required: the URL to a working dashboard~~
~~xrf367.ad6dm.net~~

~~2) Required: the address to be added to the host file~~
~~18.216.66.72~~

~~3) Required: the hosting or sponsoring station or organization~~
~~Hosted and sysop by AD6DM for JerryNet~~

~~4) Required: the country~~
~~USA~~

~~5) Optional: the organization website if it exists~~
~~kg6hqd.us~~

~~6) Optional: any title or other description for the reflector~~
~~JerryNet Reflector~~

~~7) Optional: the city and/or region~~
~~N/A~~
```

~~This process can take several days to a few weeks, depending on K6KD’s availability.~~

Note that: Under this scheme there are only 1000 reflector numbers available, most of which are taken. I am not sure what the admins plan to do when all the numbers are exhausted.

Activating your XLX Reflector

When you are ready to add your reflector to the XLX registry so that all other XLX servers will see it and report its online status, do the following:

```
$CallingHome[‘Active’] = false; // xlx phone home, true or false
```

When this gets switched to **`'true'`** your reflector will start to announce it is live and show up on the reflector list.

****When this is flipped, a file called "callinghome.php" is created in /xlxd. Make sure to copy this file or change its location in the config. This hash verifies the reflector is yours. Back it up!!****

In XLX367, this configuration is buried in systemd temp subdirectories:

```
/tmp/systemd-private-9fda321b348a472582fdbc117dXXXXXX-apache2.service-aWQbB1/tmp/callinghome.php
```

Copy that file and keep it somewhere safe! If the file disappears, and it is known to do so in the tmp directory, your reflector will be listed as offline.

Thanks to N5AMD for this information.

Configuring the YSF Reflector

If you ran the multi-mode reflector script, you will have ysfreflector installed at system file location:

```
/ysfreflector
```

There is really nothing major to update here, except you must specify the Name and Description, and it must match what you will enter in the YSF Reflector registration.

Don't worry, YSF is much faster to get going than XLX or DMR approvals.

Further configurations and customizations (including logo and **`define("SHOWQRZ", "on")`**) can be set at:

```
/var/www/ysf/config/config.php
```

Running your YSFReflector

YSFReflector is managed by systemd, so you can use:

```
systemctl restart ysfreflector
```

All the usual commands are available:

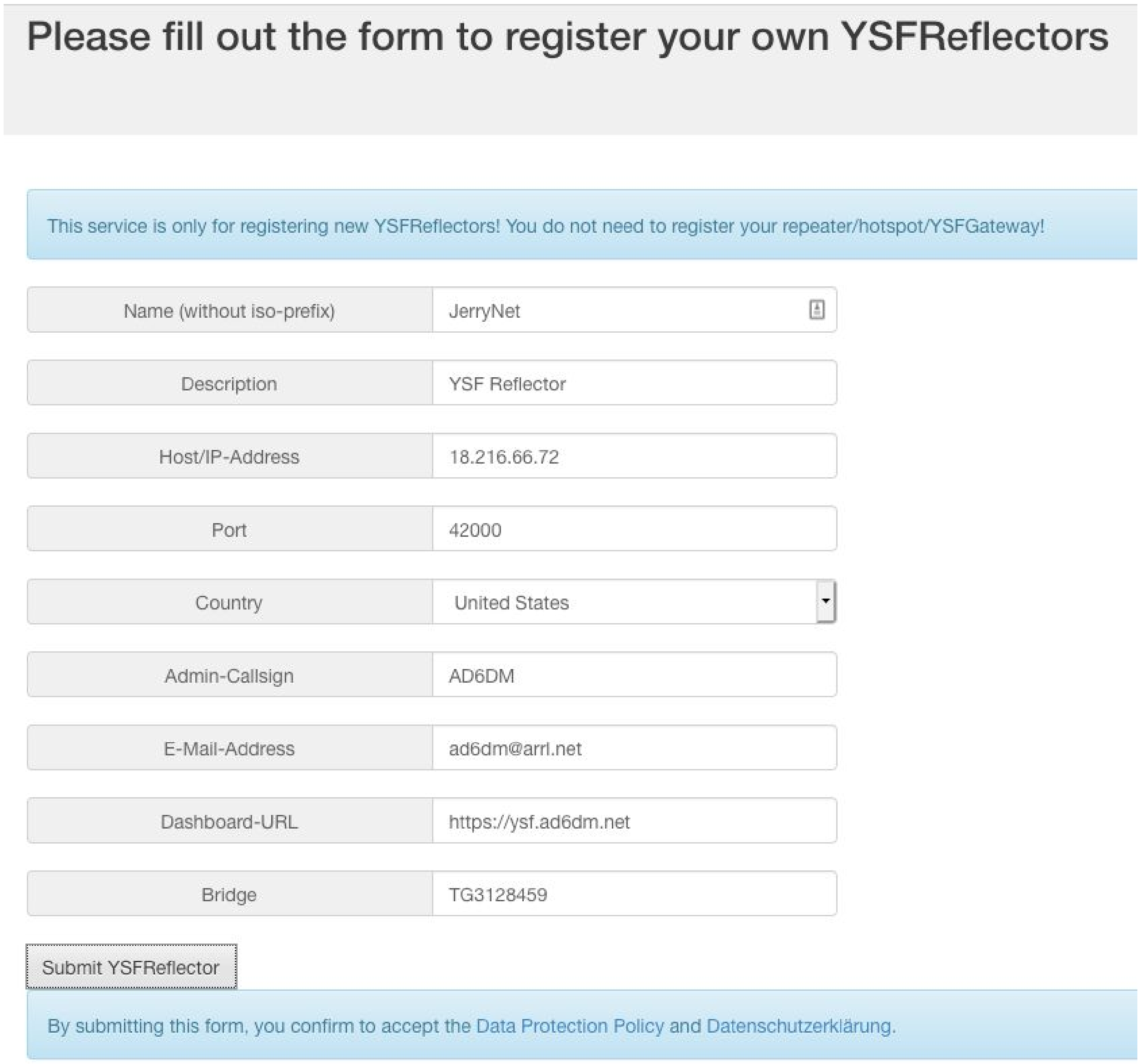
```
systemctl (start | stop | status | restart) ysfreflector
```

Registering your YSF Reflector

This process is fast. Brandmeister can take months, but YSF is quick, even minutes. By doing this, you will be assigned a YSF number by the registry maintainers. This is a number you can share with anyone to find your reflector.

Register your YSF Reflector at:

https://register.ysfreflector.de/register



Please fill out the form to register your own YSFReflectors

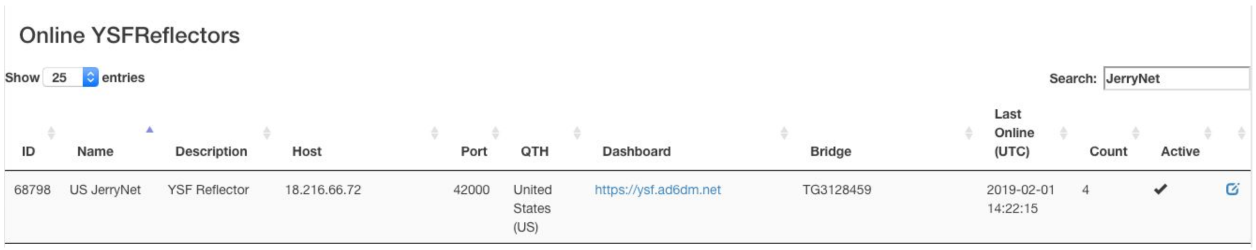
This service is only for registering new YSFReflectors! You do not need to register your repeater/hotspot/YSFGateway!

Name (without iso-prefix)	JerryNet
Description	YSF Reflector
Host/IP-Address	18.216.66.72
Port	42000
Country	United States
Admin-Callsign	AD6DM
E-Mail-Address	ad6dm@arrl.net
Dashboard-URL	https://ysf.ad6dm.net
Bridge	TG3128459

By submitting this form, you confirm to accept the Data Protection Policy and Datenschutzerklärung.

Once this information is entered, you will receive a confirmation email with a link to activate. Their system will automatically append the 2-character ISO country designation, so in our case the name became “US JerryNet”. After clicking the link to activate, the YSF

Reflector directory begins pinging your YSF reflector, and within half an hour or so, if it is configured properly, your YSF Reflector should show up for anyone who updates their Pi-Star hosts.

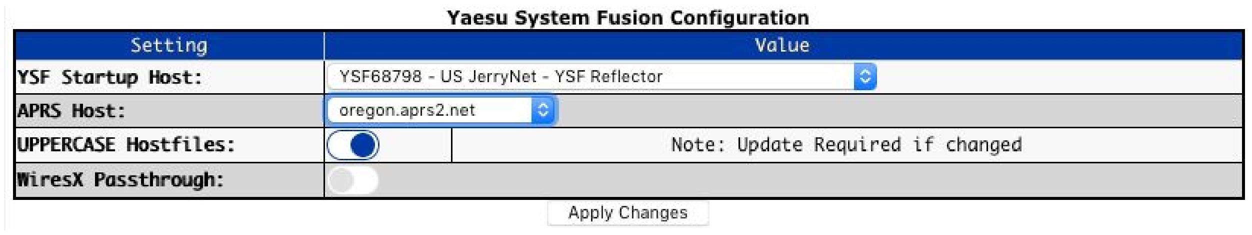


Online YSFReflectors

Show 25 entries — Search: JerryNet

ID	Name	Description	Host	Port	QTH	Dashboard	Bridge	Last Online (UTC)	Count	Active	
68798	US JerryNet	YSF Reflector	18.216.66.72	42000	United States (US)	https://ysf.ad6dm.net	TG3128459	2019-02-01 14:22:15	4	✓	

Pi-Star hotspots also update on a regular interval, so even if people don't manually update their hotspots, the settings will eventually be downloaded. Give it a day or two for most users.



Yaesu System Fusion Configuration

Setting	Value
YSF Startup Host:	YSF68798 - US JerryNet - YSF Reflector
APRS Host:	oregon.aprs2.net
UPPERCASE Hostfiles:	● Note: Update Required if changed
WiresX Passthrough:	

Configuring YSF2DMR for Initial YSF to DMR Use Only

Note here that while you wait for the XLX reflector to be approved and added to the directory, you won't be able to link the YSF reflector to your new XLX reflector. In the meantime, you can test and operate 2-mode cross-linking with Yaesu C4FM to DMR by pointing YSF2DMR directly to Brandmeister in your own configuration. This direct connection is almost exactly how pi-star hotspots do YSF2DMR allowing individual Yaesu Fusion users to be heard by DMR talkgroups.

Note also: Under this temporary setup, any unidentified YSF callsign that doesn't have a DMR ID shows up as a the YSF2DMR owner's original ID (i.e. your, the admin's, ID). And vice versa.

/ysf2dmr/YSF2DMR.ini (before XRef Directory approval and addition)

```
[Info]
RXFrequency=445800000
TXFrequency=445800000
Power=1
Latitude=38.048260
Longitude=-121.351670
```

```
Height=0
Location="Cleveland, OH"
Description=Multi-Mode Repeater
URL=http://kg6hqd.us

[YSF Network]
Callsign=AD6DM
Suffix=ND
#Suffix=RPT
DstAddress=127.0.0.1
DstPort=42000
LocalAddress=127.0.0.1
LocalPort=42013
EnableWiresX=1
RemoteGateway=0
HangTime=1000
Daemon=0

[DMR Network]
Id=1107078
#XLXFile=/ysf2dmr/XLXHosts.txt
#XLXReflector=367
#XLXModule=A
StartupDstId=3128459
# For TG call: StartupPC=0
StartupPC=0
Address=74.91.114.19 #IP of BM 3102
Port=62031 #Port of BM 3102
#Port=62030 #Port of your localhost
#Local=62031
Jitter=500
EnableUnlink=1
TGUnlink=4000
PCUnlink=0
# Local=62032
Password=passw0rd #Password of BM 3102
#Password=PASSWORD #Default password of your local XLX server
# Options=
TGListFile=/ysf2dmr/TGList-DMR.txt #Should only contain target TG
```

```
Debug=0

[DMR Id Lookup]
File=/ysf2dmr/DMRIds.dat
Time=24

[Log]
# Logging levels, 0=No logging
DisplayLevel=1
FileLevel=1
FilePath=.
FileRoot=YSF2DMR

[aprs.fi]
Enable=0
AprsCallsign=AD6DM
# Server=noam.aprs2.net
Server=oregon.aprs2.net
Port=14580
Password=18914
APIKey=Apikey
Refresh=240
Description=YSF2DMR
```

This mode will work while you are waiting for the other parts to get approved. *You will have to reconfigure this when your XLX is up and again later when Brandmeister has allowed you to interlink your XLX reflector with one of their master servers/talkgroups*. Those steps are outlined later in this document.

Running and Restarting YSF2DMR

YSF2DMR is not very well developed as of this writing. It must be run manually, and you can optionally create a startup script of your own, should the server unexpectedly reboot.

Run the following in *gnu screen*, or as a background process appending "&" (note the PID if you run it in the background so you can kill the process later to stop ysf2dmr).

```
cd /ysf2dmr

./YSF2DMR YSF2DMR.ini
```

If this needs to be restarted, re-attach the screen, stop with CTRL-C, and run it again.

Requesting XLX Interlink with Brandmeister

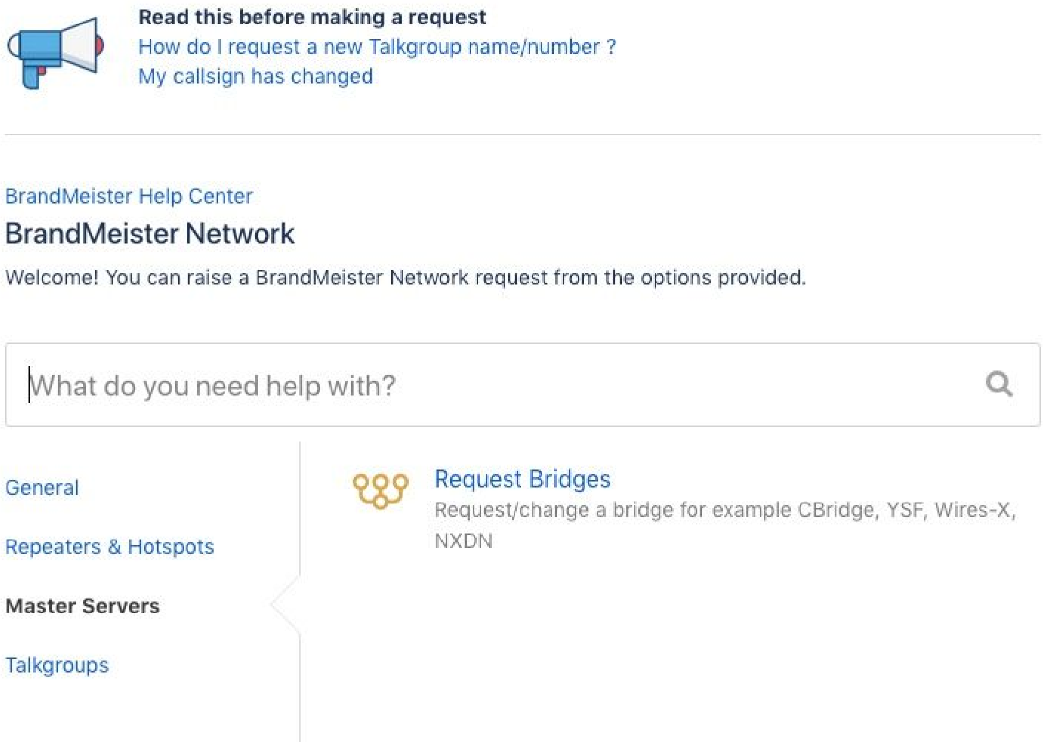
(I add this step here because it takes the longest, and you might as well start the process right away. You will use XLX interlink in Phase 3.)

Brandmeister is the main interconnected “legacy” DMR system that interlinks many DMR Masters to allow for common talkgroups that everyone can program into their radios. Other DMR formats include DMR+, a newer interlinking system, and XLX DMR, which comes with the XLX multi-protocol reflector by default.

However, linking to DMR+ and XLX DMR is not as intuitive as using Brandmeister, so to maintain compatibility it’s probably best to stick with Brandmeister as much as possible. (For more information on using DMR+ and XLX DMR, check out this great write-up by K6KD:
https://groups.io/g/KingsOfDigital/files/XLX%20XRF%20DMRGateway:Pi-Star%20060818.pdf). (Requires groups.io account and joining the Kings of Digital group.)

In order to link your XLX reflector to a BM master you must submit a “Request Bridge” support ticket under the Brandmeister Support page.
https://support.brandmeister.network/login.jsp

It is under Brandmeister Network -> Master Servers -> Request Bridges.



Read this before making a request

How do I request a new Talkgroup name/number ?

My callsign has changed

BrandMeister Help Center

BrandMeister Network

Welcome! You can raise a BrandMeister Network request from the options provided.

What do you need help with?

Request Bridges

Request/change a bridge for example CBridge, YSF, Wires-X, NXDN

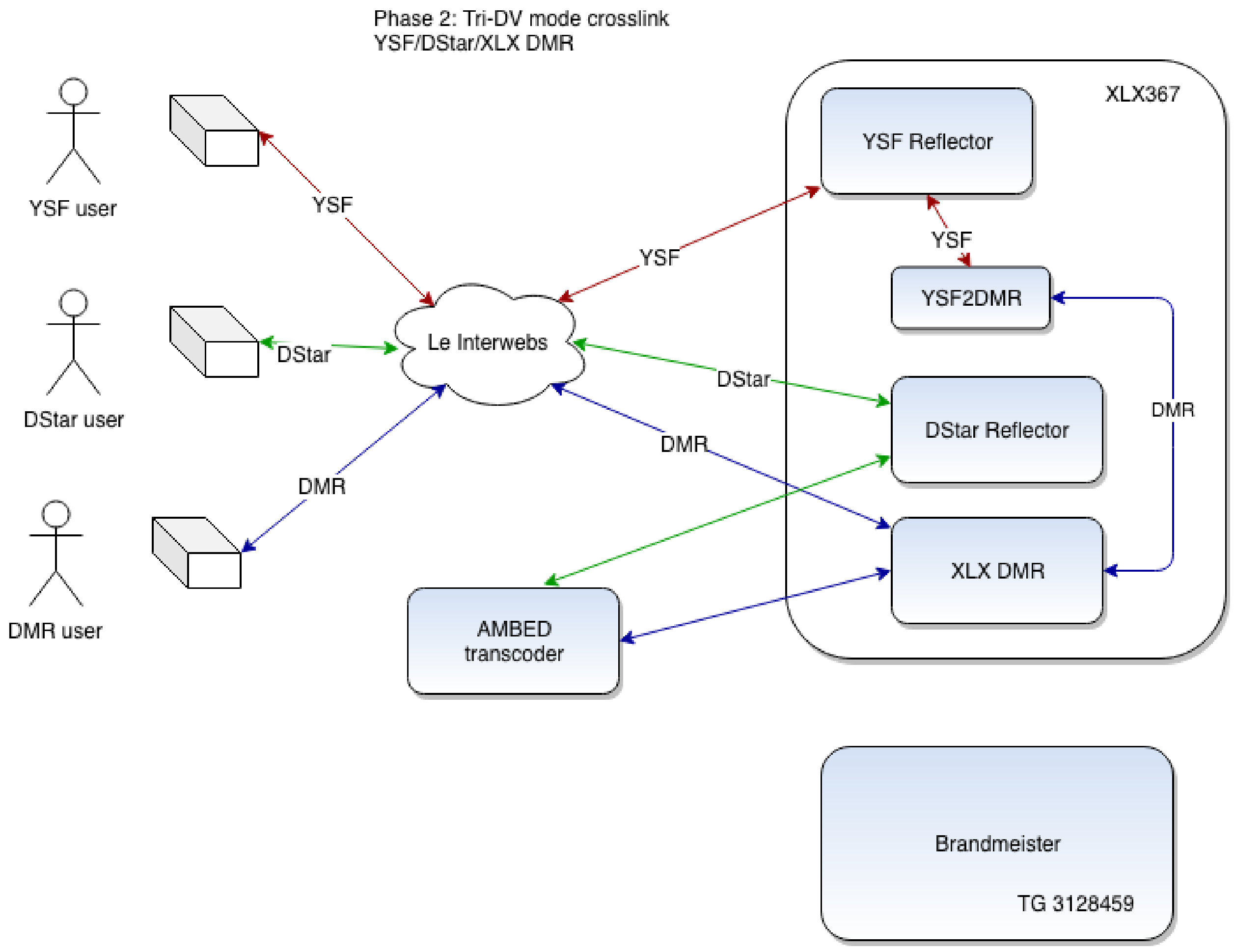
Submit a new ticket there with all the information about your XLX Reflector, including the need for an XLX Interlink, the BM Master (e.g. 3102) that you want to link to, the target BM talkgroup (in our case *TG 3128459*), and which modules you want to link to Brandmeister. As of this writing, this process takes a few months. :(

Phase 2: YSF to XLX DMR with DStar Transcoding (Tri-mode crosslink)

In this phase while you wait for the Brandmeister interlink request, you *can still have a 3-mode crosslink between DMR, DStar, and YSF*.

However, this will mean you will **lose connectivity to the Brandmeister talkgroup**. Depending on your users, you can choose to stay in Phase 1 until Brandmeister gets to your support ticket then skip to Phase 3. But if you can instruct your users on how to connect to XLX DMR, this is a great way to get all 3 radios talking under one reflector.

Diagram of Phase 2



Configuring YSF2DMR link to Your Own XLX

To link YSF and DMR to your XLX Reflector, modify the YSF2DMR.ini as follows:

```
[DMR Network]

Id=YourDMRID

XLXFile=/ysf2dmr/XLXHosts.txt

XLXReflector=XXX # your reflector number

XLXModule=A # The module you want to link to

StartupDstId=4001 # 4001 corresponds to A


#StartupDstId=3128459 # this is now commented out
```

```
# For TG call: StartupPC=0

StartupPC=0

Address=18.216.66.72 # update to the IP of your XLX reflector

Port=62030 # this should be default of your XLX reflector
```

Now you are ready to implement transcoding from DStar to your XLX DMR which will be heard by YSF users.

AMBED Server for DStar Transcoding

The DMR and DStar aspects built into an XLX reflector coexist without interfering with each other, and only by adding ambed software and transcoding channels can they hear each other.

The server where ambed runs requires physical access because you need to install AMBE3000 hardware either on USB ports or with another type of AMBE3000 board of some sort. Therefore it is difficult to do in the cloud. Many have installed it on a Raspberry Pi 3 in a home Internet network.

This goes against the high-availability introduction of this document, so if you can find a way to keep the AMBED installation in a colo or in the cloud that would be ideal! Remember: You do not want high network latency between the XLX server and the ambed server, as this could cause transcoding to fail. It is best that they are not geographically very distant from each other, that you have fast Internet, and do not use wifi for the ambed server (preferring a wired connection).

Prepping and Installing AMBE3000 devices

You need at least a pair of AMBE 3000 devices, *one for DStar to DMR* transcoding, and *one for DMR to DStar transcoding* back. The ones I used are called ThumbDV™ AMBE3000 Digital Voice Vocoder on USB, and can be found at NW Digital Radio:

http://nwdigitalradio.com/product/thumbdv/

A pair of ThumbDV's provides these **two** transcoding channels, which means only **one** module on your reflector (e.g. Module A, so in our case: XRF367A) should utilize the transcoding capability. Each additional pair of AMBE3000 devices provides bidirectional transcoding for one more module. (In theory you can use more than one module with only one pair of ThumbDV's, but that only invites unavailability and audio loss if your reflector gets too many concurrent audio streams.)

Other AMBE devices provide more channels, such as the DV Mega AMBE3003 DV USB Stick-- a pair of which would provide 3 transcoding channels each USB stick, and a total of 4 channels that could run concurrently. Only 1 AMBE3003 USB device would be needed to provide 2 concurrent streams if you wanted to avoid buying a pair of USB sticks.

The XLX367 transcoding server: Raspberry 3B+ on wired network with 2 ThumbDV's.

You need to "program" each of the ThumbDVs to have the right device name. This is a limitation of the *ambed* software, so any device manufacturer name that is not "USB-3000" will presumably cause the device to not work. (I did not test this claim.)

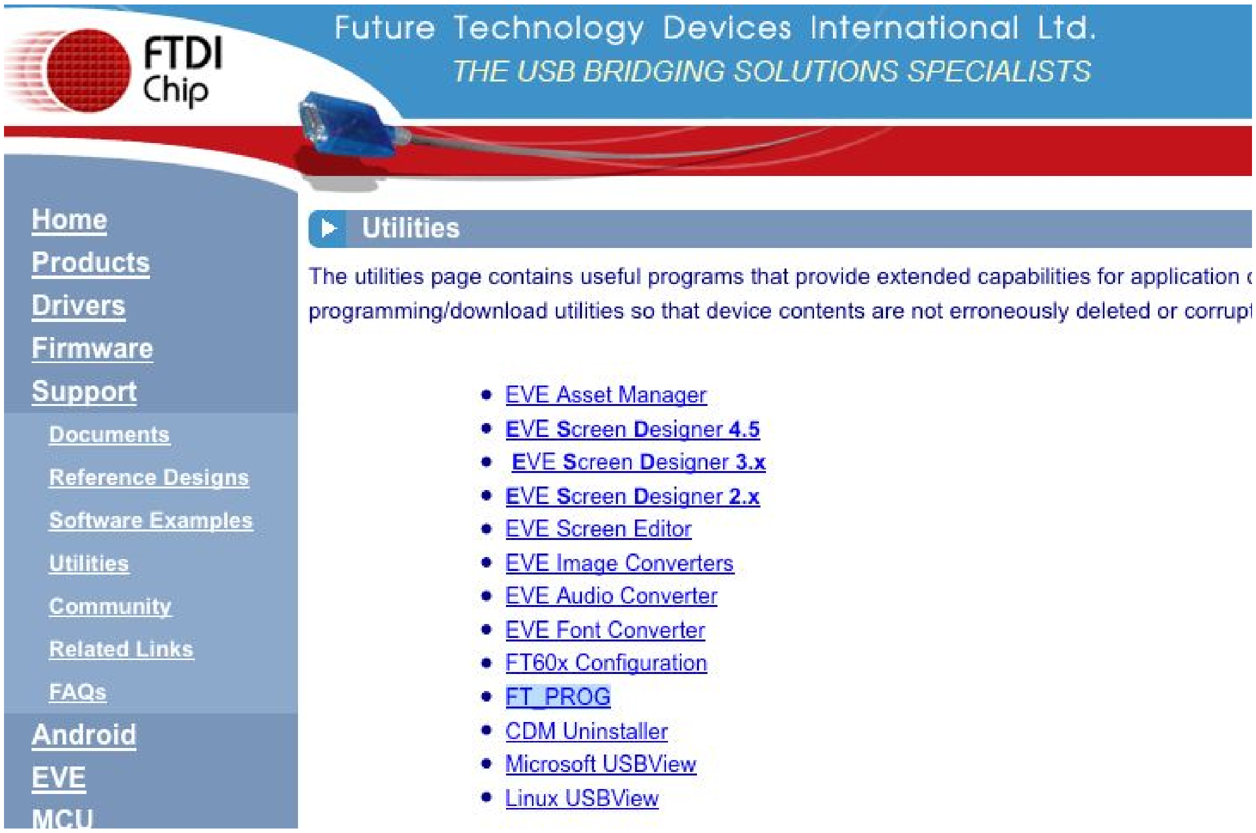
I copied this information out of the source document (with some minor copyedits):

http://hamradio.dip.jp/ja3gqj/ambed_server_myexp.pdf

Change ThumbDV Product Description (Deprecated as of April 2019)

***Note**: A few who have followed this document and created their own transcoding servers have reported that this step is not necessary. It seems* `ambed` *can find AMBE3000 USB sticks without any naming changes. Examples include the AMBE3003 USB from DVMega, or the ThumbDV™ described below working fine without modified naming. Therefore this section is deprecated and **you do not need to perform** these Product Description renaming steps. **Continue to Obtaining the FTDI linux drivers for your AMBE3000 devices**.*

1. ~~Devices connected to ambed server via USB are based on USB-3000.~~
   ~~ThumbDV with similar functions can be substituted, but in order to substitute, it is necessary to change "Product Description" of ThumbDV to "USB-3000" character string.~~
2. ~~In order to change "Product Description" of ThumbDV, it is necessary to install "FT Prog" tool on Windows PC.~~
3. ~~Obtaining "FT Prog" tool~~
   ~~Click "FT Prog" at~~
   ~~http://www.ftdichip.com/Support/Utilities.htm~~

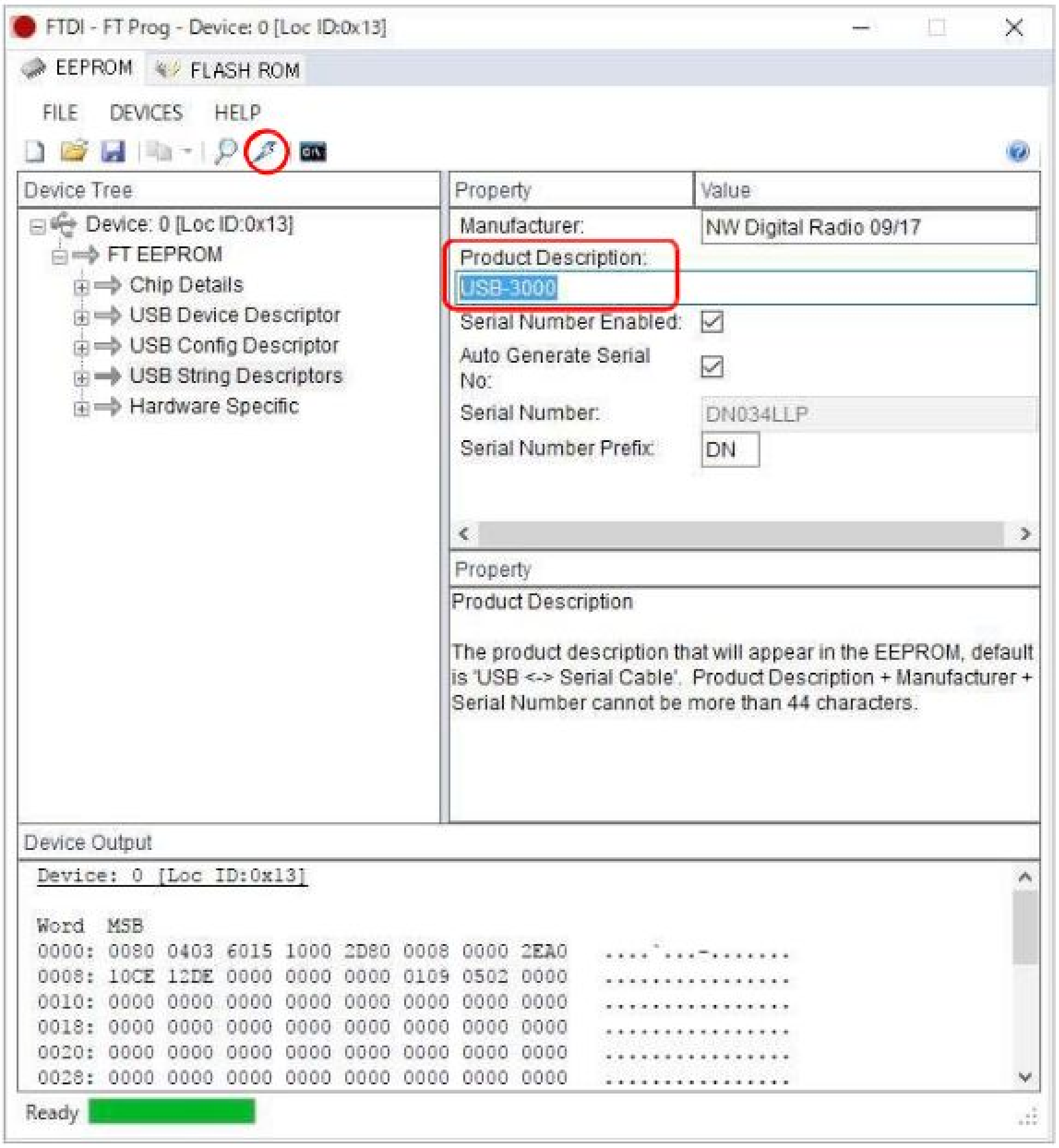


The utilities page contains useful programs that provide extended capabilities for application
programming/download utilities so that device contents are not erroneously deleted or corrupt

- EVE Asset Manager
- EVE Screen Designer 4.5
- EVE Screen Designer 3.x
- EVE Screen Designer 2.x
- EVE Screen Editor
- EVE Image Converters
- EVE Audio Converter
- EVE Font Converter
- FT60x Configuration
- FT_PROG
- CDM Uninstaller
- Microsoft USBView
- Linux USBView

~~Change the "Product Description"~~

1. ~~Connect ThumbDV to USB port of Windows PC.~~
2. ~~Execute the "FT Prog" tool.~~
3. ~~Press the "F5" key of keyboard to read the DEVICE information.~~
4. ~~Click "USB String Descriptors" in "Device Tree".~~
5. ~~"Product Description" shows "ThumbDV". Change it here.~~
6. ~~Overwrite "USB String Descriptors" from "ThumbDV" to "USB-3000".~~
7. ~~Click the icon of the lightning mark.~~



Property	Value
Manufacturer:	NW Digital Radio 09/17
Product Description:	USB-3000
Serial Number Enabled:	✓
Auto Generate Serial No:	✓
Serial Number:	DN034LLP
Serial Number Prefix:	DN

Property

Product Description

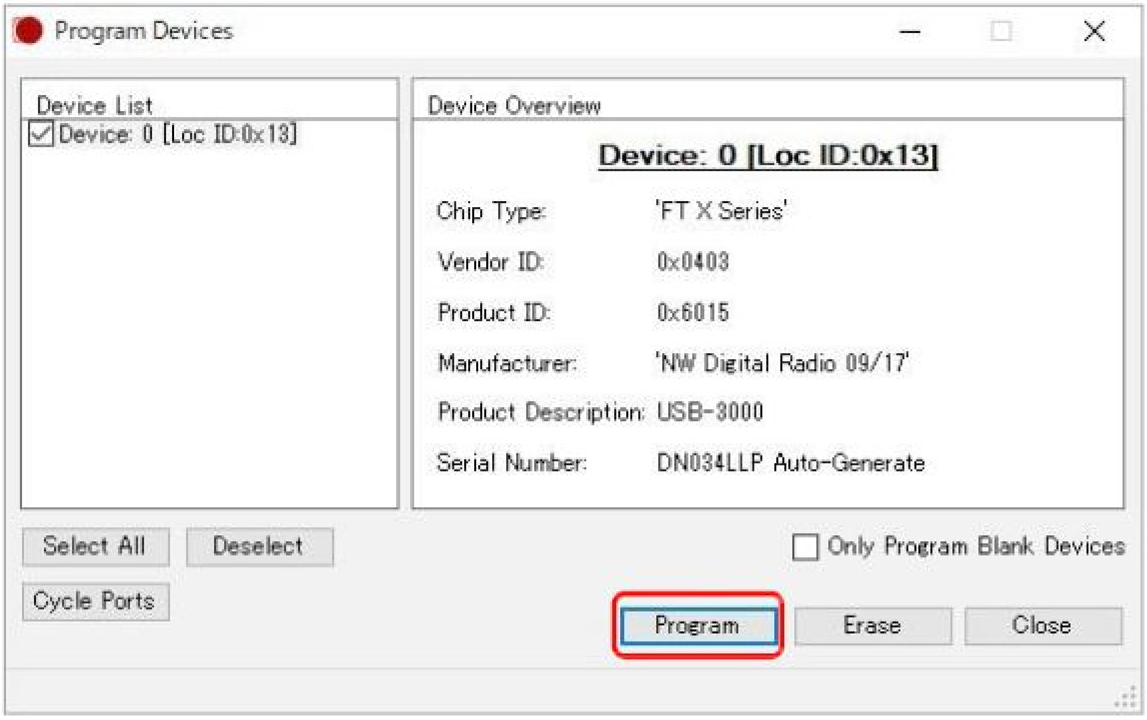
The product description that will appear in the EEPROM, default is 'USB <-> Serial Cable'. Product Description + Manufacturer + Serial Number cannot be more than 44 characters.

Device Output

```
Device: 0 [Loc ID:0x13]

Word  MSB
0000: 0080 0403 6015 1000 2D80 0008 0000 2EA0   ....`...-.......
0008: 10CE 12DE 0000 0000 0000 0109 0502 0000   ................
0010: 0000 0000 0000 0000 0000 0000 0000 0000   ................
0018: 0000 0000 0000 0000 0000 0000 0000 0000   ................
0020: 0000 0000 0000 0000 0000 0000 0000 0000   ................
0028: 0000 0000 0000 0000 0000 0000 0000 0000   ................
```

Ready

~~8. Click "Program".~~


Program Devices

Device List

☑ Device: 0 [Loc ID:0x13]

Device Overview

Device: 0 [Loc ID:0x13]

Chip Type:	'FT X Series'
Vendor ID:	0x0403
Product ID:	0x6015
Manufacturer:	'NW Digital Radio 09/17'
Product Description:	USB-3000
Serial Number:	DN034LLP Auto-Generate

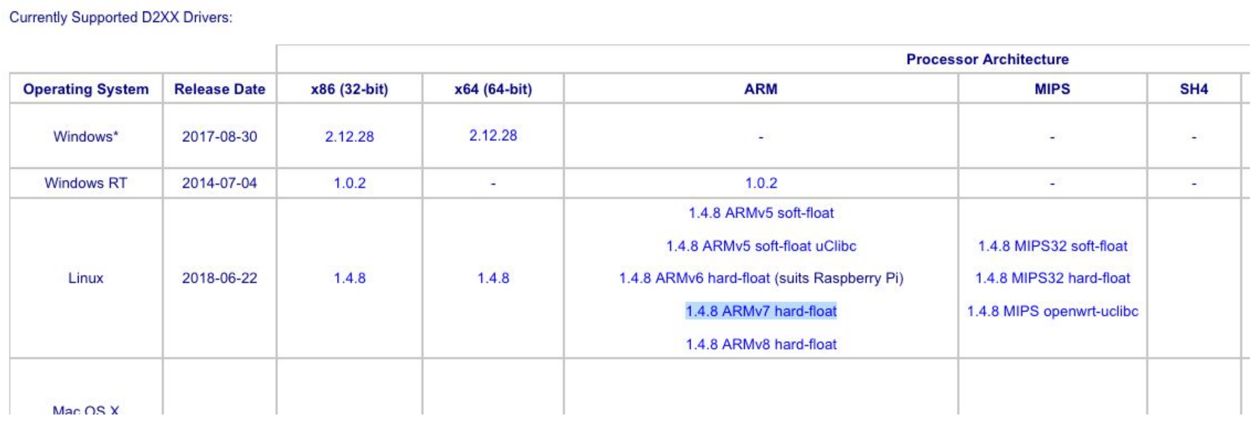
Obtain the FTDI linux drivers for your AMBE3000 devices

These steps you run on the Raspberry Pi 3 that you will be connecting your ThumbDVs to.

Drivers can be found at:

https://www.ftdichip.com/Drivers/D2XX.htm

The drivers you want are ***1.4.8 ARMv7 hard-float*** or whatever is current for ARMv7. I read the above document citing this specific version for Raspberry Pi 3, even though at the time of that document's writing, Pi 3 had an ARMv8 chip. I did not question this, and feel free to try the v8 driver if you want, but it worked well with ***1.4.8 ARMv7 hard float*** for the XLX367 reflector install.



Currently Supported D2XX Drivers:

		Processor Architecture				
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	ARM	MIPS	SH4
Windows*	2017-08-30	2.12.28	2.12.28	-	-	-
Windows RT	2014-07-04	1.0.2	-	1.0.2	-	-
Linux	2018-06-22	1.4.8	1.4.8	1.4.8 ARMv5 soft-float 1.4.8 ARMv5 soft-float uClibc 1.4.8 ARMv6 hard-float (suits Raspberry Pi) 1.4.8 ARMv7 hard-float 1.4.8 ARMv8 hard-float	1.4.8 MIPS32 soft-float 1.4.8 MIPS32 hard-float 1.4.8 MIPS openwrt-uclibc	
Mac OS X						

Installing ambed Software

After configuring the AMBE3000 chips, put them aside and you will now install *ambed* software on the Pi.

Login to the pi and switch user to root, then perform the following:

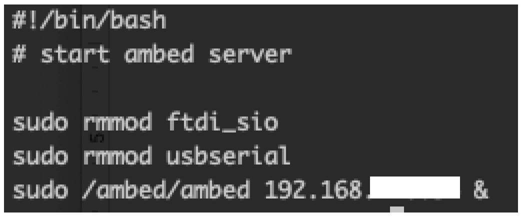
```
# git clone https://github.com/LX3JL/xlxd.git


Cloning into ‘xlxd’…

remote: Counting objects: 1683, done.

remote: Compressing objects: 100% (12/12), done.

remote: Total 1683 (delta 2), reused 2 (delta 0), pack-reused 1671

Receiving objects: 100% (1683/1683), 971.52 KiB | 0 bytes/s, done.

Resolving deltas: 100% (1055/1055), done.


# cd xlxd


# cd ambed
# wget
http://www.ftdichip.com/Drivers/D2XX/Linux/libftd2xx-arm-v7-hf-1.4.8.t
gz

# tar xvfz libftd2xx-arm-v7-hf-1.4.8.tgz

# cd release

# cd build

# cp libftd2xx.* /usr/local/lib

# chmod 0755 /usr/local/lib/libftd2xx.so.1.4.8

# ln -sf /usr/local/lib/libftd2xx.so.1.4.8 /usr/local/lib/libftd2xx.so

# cd ../../xlxd/ambed

# make

g++ -c -std=c++11 -pthread cusb3003df2etinterface.cpp -o
cusb3003df2etinterface.o

g++ -c -std=c++11 -pthread cpacket.cpp -o cpacket.o

g++ -c -std=c++11 -pthread ccallsign.cpp -o ccallsign.o

g++ -c -std=c++11 -pthread cudpsocket.cpp -o cudpsocket.o

g++ -c -std=c++11 -pthread cambepacket.cpp -o cambepacket.o

g++ -c -std=c++11 -pthread cambeserver.cpp -o cambeserver.o
```

```
g++ -c -std=c++11 -pthread cstream.cpp -o cstream.o

g++ -c -std=c++11 -pthread cusb3000interface.cpp -o
cusb3000interface.o

g++ -c -std=c++11 -pthread cvocodecchannel.cpp -o cvocodecchannel.o

g++ -c -std=c++11 -pthread cusb3003hrinterface.cpp -o
cusb3003hrinterface.o

g++ -c -std=c++11 -pthread cusb3xxxinterface.cpp -o
cusb3xxxinterface.o

g++ -c -std=c++11 -pthread ctimepoint.cpp -o ctimepoint.o

g++ -c -std=c++11 -pthread cvocodecinterface.cpp -o
cvocodecinterface.o

g++ -c -std=c++11 -pthread cftdidevicedescr.cpp -o cftdidevicedescr.o

g++ -c -std=c++11 -pthread cusb3003interface.cpp -o
cusb3003interface.o

g++ -c -std=c++11 -pthread cip.cpp -o cip.o

g++ -c -std=c++11 -pthread cvocodecs.cpp -o cvocodecs.o

g++ -c -std=c++11 -pthread cvoicepacket.cpp -o cvoicepacket.o

g++ -c -std=c++11 -pthread cpacketqueue.cpp -o cpacketqueue.o

g++ -c -std=c++11 -pthread cbuffer.cpp -o cbuffer.o

g++ -c -std=c++11 -pthread ccontroller.cpp -o ccontroller.o

g++ -c -std=c++11 -pthread main.cpp -o main.o

g++ -std=c++11 -pthread cusb3003df2etinterface.o cpacket.o ccallsign.o
cudpsocket.o cambepacket.o cambeserver.o cstream.o cusb3000interface.o
cvocodecchannel.o cusb3003hrinterface.o cusb3xxxinterface.o
ctimepoint.o cvocodecinterface.o cftdidevicedescr.o
cusb3003interface.o cip.o cvocodecs.o cvoicepacket.o cpacketqueue.o
cbuffer.o ccontroller.o main.o -lftd2xx -Wl,-rpath,/usr/local/lib -o
ambed


# make install
# cd /ambed
```

```
# chmod 777 run
```

Edit the run file. If AMBED is running on the same machine as xlxd, use the default 127.0.0.1.

Otherwise, set the Raspberry Pi's IP address.



```
#!/bin/bash
# start ambed server

sudo rmmod ftdi_sio
sudo rmmod usbserial
sudo /ambed/ambed 192.168[redacted] &
```

Running and restarting with the AMBE3000 devices

ambed is not a very robust piece of software. It does not run as a service, and it stays tied to the terminal you are using. I recommend running it within *gnu screen*.

Running the software is as simple as doing:

```
# cd /ambed

# ./run
```

Restarting the software is more tricky. If the ambed process exits for any reason, including you manually stopping it, it will not be able to find the USB devices when you run it again.

You need to reboot the Pi and run it again from a new session to restart ambed.

Linking XLX to AMBED

On the XLX server, edit the /etc/init.d/xlxd file:

```
ARGUMENTS="XLX099 212.237.59.103 127.0.0.1"
```

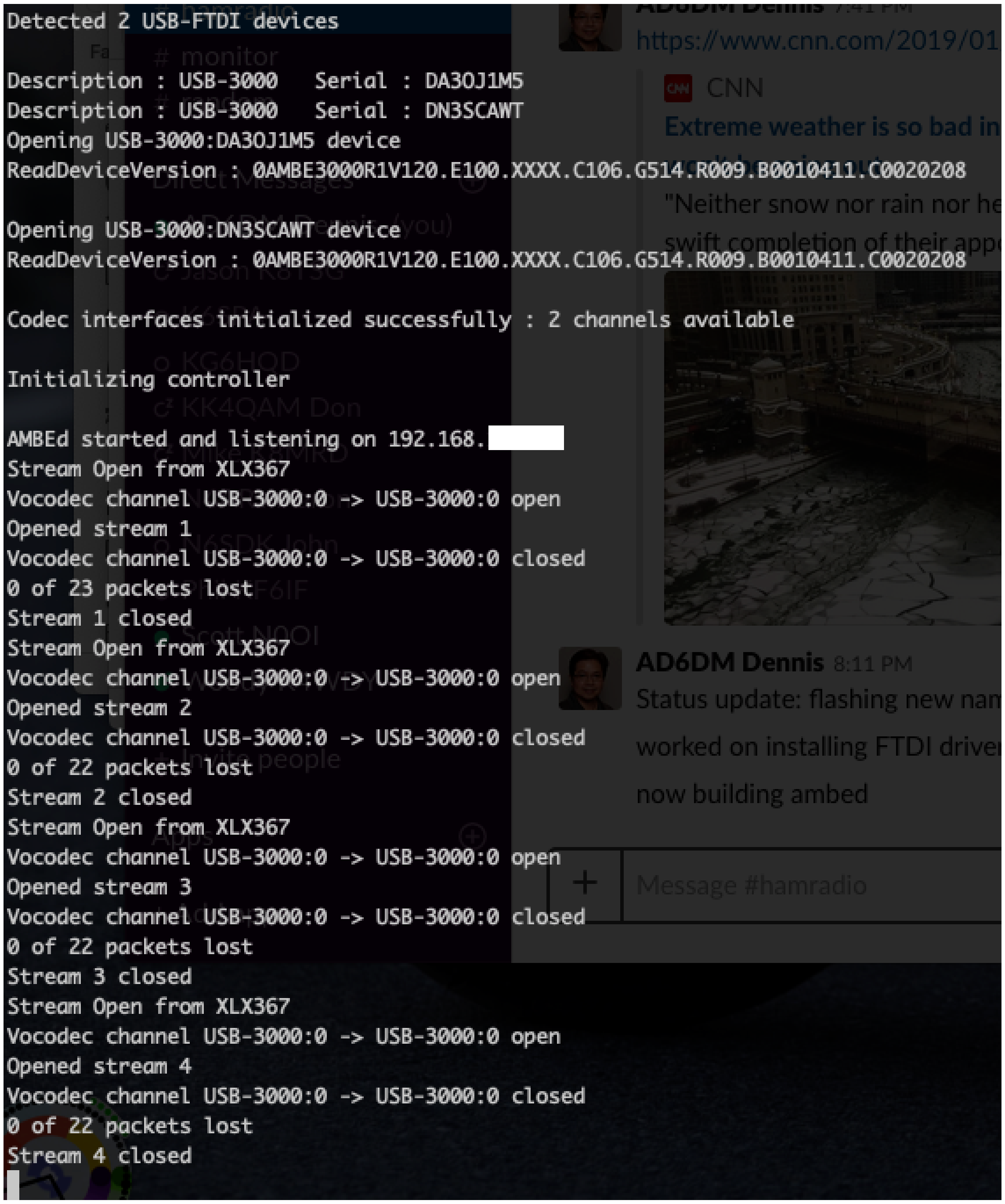
Where "`127.0.0.1`" is the IP address of your AMBED server e.g. 192.168.1.10. If you're doing this at home, for example, on a Raspberry Pi, make sure to open up ports

10100-10199 on your home router and port forward them to the internal network IP address of your Pi.

Now you can run the ambed software:

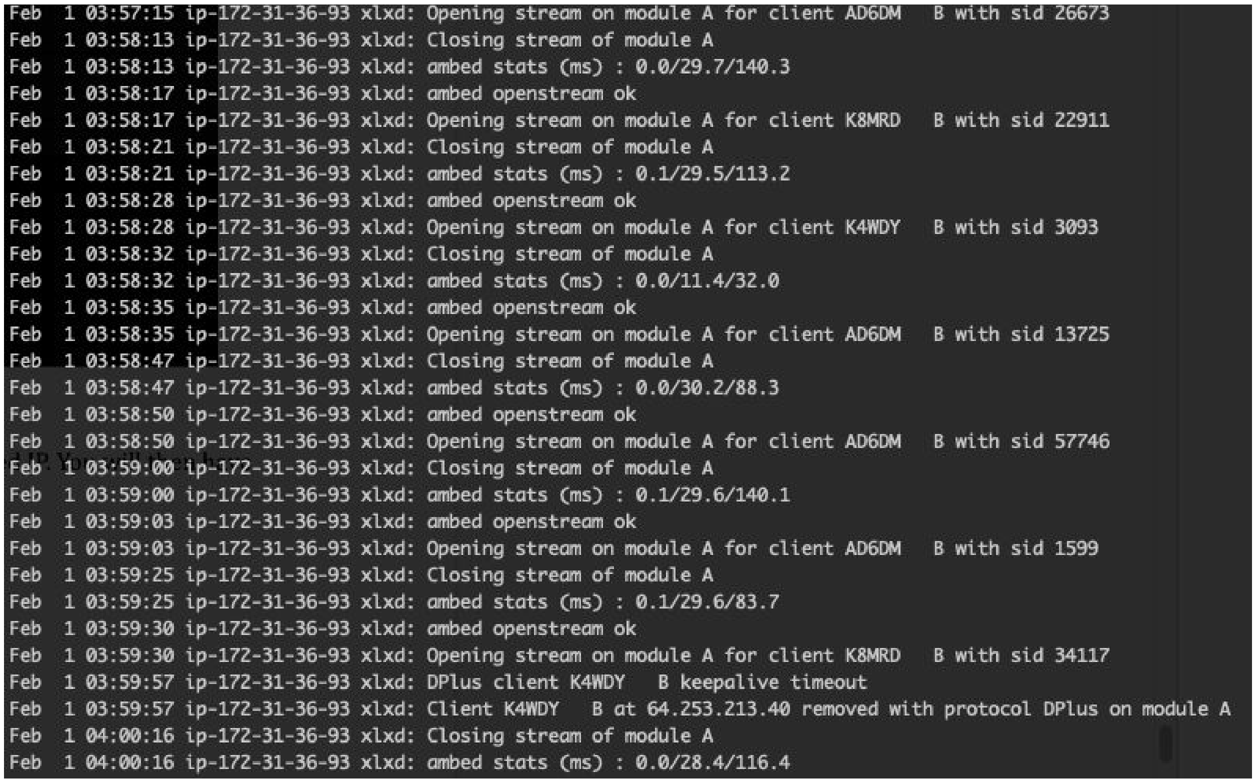
```
# cd /ambed
```

```
# ./run
```



```
Detected 2 USB-FTDI devices

Description : USB-3000   Serial : DA3OJ1M5
Description : USB-3000   Serial : DN3SCAWT
Opening USB-3000:DA3OJ1M5 device
ReadDeviceVersion : 0AMBE3000R1V120.E100.XXXX.C106.G514.R009.B0010411.C0020208

Opening USB-3000:DN3SCAWT device
ReadDeviceVersion : 0AMBE3000R1V120.E100.XXXX.C106.G514.R009.B0010411.C0020208

Codec interfaces initialized successfully : 2 channels available

Initializing controller

AMBEd started and listening on 192.168.[redacted]
Stream Open from XLX367
Vocodec channel USB-3000:0 -> USB-3000:0 open
Opened stream 1
Vocodec channel USB-3000:0 -> USB-3000:0 closed
0 of 23 packets lost
Stream 1 closed
Stream Open from XLX367
Vocodec channel USB-3000:0 -> USB-3000:0 open
Opened stream 2
Vocodec channel USB-3000:0 -> USB-3000:0 closed
0 of 22 packets lost
Stream 2 closed
Stream Open from XLX367
Vocodec channel USB-3000:0 -> USB-3000:0 open
Opened stream 3
Vocodec channel USB-3000:0 -> USB-3000:0 closed
0 of 22 packets lost
Stream 3 closed
Stream Open from XLX367
Vocodec channel USB-3000:0 -> USB-3000:0 open
Opened stream 4
Vocodec channel USB-3000:0 -> USB-3000:0 closed
0 of 22 packets lost
Stream 4 closed
```

Restart xlxd on your XLX server, and see it connect to your ambed IP. You will then have DStar to DMR transcoding!

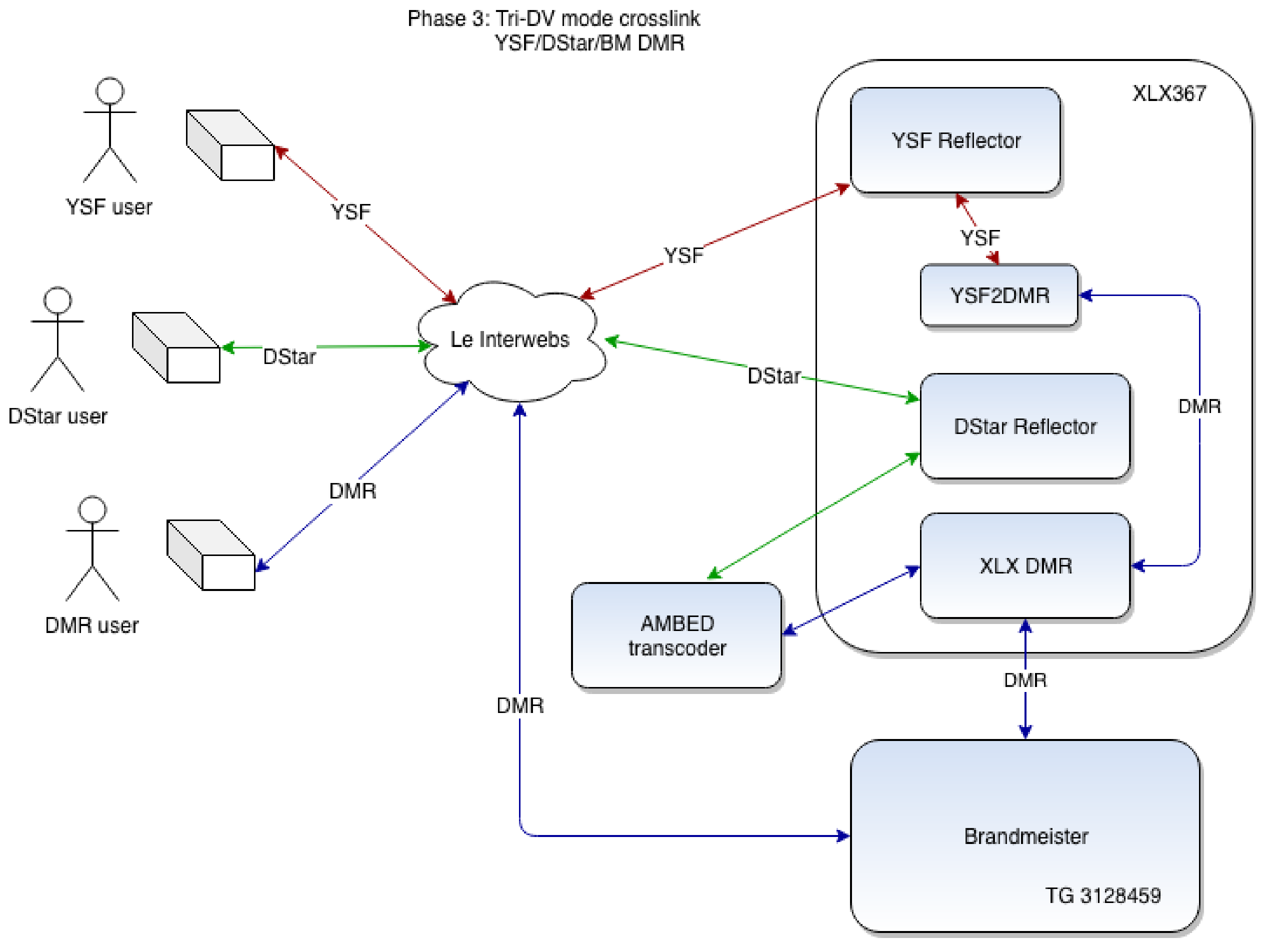


```
Feb  1 03:57:15 ip-172-31-36-93 xlxd: Opening stream on module A for client AD6DM   B with sid 26673
Feb  1 03:58:13 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:58:13 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.0/29.7/140.3
Feb  1 03:58:17 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:58:17 ip-172-31-36-93 xlxd: Opening stream on module A for client K8MRD   B with sid 22911
Feb  1 03:58:21 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:58:21 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.1/29.5/113.2
Feb  1 03:58:28 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:58:28 ip-172-31-36-93 xlxd: Opening stream on module A for client K4WDY   B with sid 3093
Feb  1 03:58:32 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:58:32 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.0/11.4/32.0
Feb  1 03:58:35 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:58:35 ip-172-31-36-93 xlxd: Opening stream on module A for client AD6DM   B with sid 13725
Feb  1 03:58:47 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:58:47 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.0/30.2/88.3
Feb  1 03:58:50 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:58:50 ip-172-31-36-93 xlxd: Opening stream on module A for client AD6DM   B with sid 57746
Feb  1 03:59:00 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:59:00 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.1/29.6/140.1
Feb  1 03:59:03 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:59:03 ip-172-31-36-93 xlxd: Opening stream on module A for client AD6DM   B with sid 1599
Feb  1 03:59:25 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 03:59:25 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.1/29.6/83.7
Feb  1 03:59:30 ip-172-31-36-93 xlxd: ambed openstream ok
Feb  1 03:59:30 ip-172-31-36-93 xlxd: Opening stream on module A for client K8MRD   B with sid 34117
Feb  1 03:59:57 ip-172-31-36-93 xlxd: DPlus client K4WDY   B keepalive timeout
Feb  1 03:59:57 ip-172-31-36-93 xlxd: Client K4WDY   B at 64.253.213.40 removed with protocol DPlus on module A
Feb  1 04:00:16 ip-172-31-36-93 xlxd: Closing stream of module A
Feb  1 04:00:16 ip-172-31-36-93 xlxd: ambed stats (ms) : 0.0/28.4/116.4
```

Phase 3: Integration with Brandmeister for True 3-mode Crosslink

After Brandmeister responds to your support ticket (months after you requested it), you can now perform the last tiny step to link a talkgroup of your choosing to your XLX reflector.

Diagram of Phase 3



Everyone talks to everyone! Woohoo!

Setting up XLX Interlink with Brandmeister

I place this section last because this is likely the last step due to Brandmeister admins taking so long to address submitted support tickets. You will likely wait months for any XLX interlink request to be answered by Brandmeister server owners.

***Note**: For XLX367, the request took exactly 2 months to be approved by Brandmeister admins and enabled on BM3108.*

When Brandmeister has confirmed their setup is ready, you can modify your xlx configuration to complete the interlink with Brandmeister Master.

```
# cd /xlxd
```

```
# vim xlxd.interlink
```

(add this line):

```
BM3108 3108.repeater.net A
```

First part (`BM3108`) is the name that will be displayed in the dashboard, can be anything, but it is most helpful if descriptive. For this case is it the display name for Brandmeister Master 3108.

Second parameter is either the IP address or FQDN of the Brandmeister Master, so here in the example above you could put `3108.repeater.net` (or alternately, `64.94.238.196`).

Third parameter is the list of modules you want to link to Brandmeister. In my case I am only use Module A for interlinking to Brandmeister. However you can put something like `ABC` to link those three modules to Brandmeister. (Your initial Bridge Request ticket should have specified the TGs you wanted those modules to link to.)

Once this is saved xlxd will load the interlink file automatically. You are LIVE!

Congratulations, you are now part of the XReflector transcoding digital ham radio network!

Project Cost

A quick summary of monetary costs relating to this project:

- AWS t2.micro instance for XLX Reflector server: ~$10/month.
- Raspberry Pi 3B+ with case: ~$40.00.
- NW Digital Radio ThumbDV USB AMBE3000 (x2): $242.56.
    - *Special thanks to Don KK4QAM for donating one of the two to the JerryNet Reflector project!*
- Several days of your time configuring and testing: $priceless.
- Any radios you want to get to test the different digital voice modes: $endless.

References

An AMBED Server Experience (document)
http://hamradio.dip.jp/ja3gqj/ambed_server_myexp.pdf

Brandmeister Support
You will almost certainly need to interact with these admins if you are using BM DMR with your XReflector.
https://support.brandmeister.network/secure/Dashboard.jspa

How to Link DMR and DStar by AF9W Bob Stephens
Informative and helpful for understanding how these dv modes work, but not quite relevant to the above installation procedure.
http://w7hd.ddns.net/digital/DMRGateway.pdf

Procedure for Installing XLX Transcoding System by IZ7AUH
https://www.qsl.net/kb9mwr/projects/dv/multi/Procedure%20For%20Installing%20The%20XLX%20Transcoding%20System.pdf

Kings of Digital Discussion Board
Kings of Digital maintains a constellation of ham radio reflectors.
https://groups.io/g/KingsOfDigital/

Multi-Reflector Installer
Gets you started in the right direction, but far from a complete install.
https://github.com/n5amd/Multi-Reflector-Installer

XLX Brandmeister config
https://wiki.brandmeister.network/index.php/XLX

XLX DMR explained
Found this funny and interesting: A lot of diagrams, but still ends in a question.
http://www.rlx.lu/forum/xlx/212-xlx270-goes-dmr-explained.html

XLX and XRF Reflectors, DMR, and DMRGateway with Pi-Star
A very good explanation of XLX, and the different kinds of DMR.
https://groups.io/g/KingsOfDigital/files/XLX%20XRF%20DMRGateway:Pi-Star%20060818.pdf

XLX Forum
According to many, all your questions about XLX should go here, moderator approval for subscription, I've waited 2 weeks as of this writing and still not approved for viewing.
http://xlxbbs.epf.lu/

XLX Yahoo Group

A horrendously organized discussion board, but lots of useful tidbits of information.
https://groups.yahoo.com/neo/groups/xlxd-star

YSF2DMR sample config

After much searching, this page (in Thai) finally got me to figuring out how to get YSF2DMR to work with direct BM linking.
http://www.dtdxa.com/index.php/2018/02/14/ysf2dmr-installation-pi-star/

YSF2DMR source code

https://github.com/juribeparada/YSF2DMR

YSF Reflector Registry

https://register.ysfreflector.de/

Dennis AD6DM

https://ad6dm.net

Twitter @AD6DM

If you found any of this useful, consider donating to the AD6DM Tip Jar. 73!