# WinTak Notes

## Overall Design



Figure High Level Deployment



Figure Deployment Detail

The TAK communications can take place over one of two protocols:

1. Multicast: Default 239.2.3.1:6969
2. TCP Unicast

TAK clients (such as ATAK and WinTAK) can talk to each other over multicast if available. Directed messages are sent between two specific clients via TCP Unicast (via a TAK Server, or directly. I’m not sure – see <https://github.com/FreeTAKTeam/FreeTakServer/blob/master/FreeTAKServer/README.md>?)

Clients can send tactical objects directly to each other over multicast. The data model for this is described below.

Clients can also talk to each other via a TAKServer, via TCP Unicast. There is something called the “Marti” API/Protocol that is used to talk to the TAKServer. I’m not exactly sure of all the methods of this protocol.

### Data models

The ATAK/WinTAK data models are based around “Cursor on Target” (CoT) messaging.

The data models are expressed in one of two formats:

1. XML. This is the original data model expression. The various types of messages that can be sent are defined in xsds. All TAK clients and servers are required to support XML messaging. When sending an XML CoT message *you do not need to provide any special header* (as you need to provide for the Protobuf message format below). To send an XML message you merely have to create the XML and then serialize it into ascii (utf-8).
2. Protobuf: This is a newer expression of the data model. It is based around google protobufs. The advantages with protobuf are:
   1. the “on-the-wire” message is smaller. Typically you can get a 50% reduction
   2. It is faster to interpret (because the various programming languages allow “accessor methods” to the message fields (which is fast), rather than XML needing to be parsed (which is slow).

When you are sending a protobuf message then you have to provide a special 3-byte header that consists of: <magic byte><protocol><magic byte>, where <magic byte>=0xbf, and <protocol>=0x01 for protobuf. (The documentation says that <protocol>=0x00 for XML, however, if you include this header with an XML transmission then neither ATAK nor WinTAK pick it up – so don’t do it).

The WinTAK data schema is built around a small set of elements defined as an “event”. The complete xsd definition is in the “CoT Base-Event Schema (PUBLIC RELEASE).xsd”. The most important event attributes are:

1. “time” (which is the reporting time)
2. “start” (which is the time that the event will start (which can be different than the reporting time)
3. “stale” (which is the time when the information about the event is no longer valid – it’s gone stale). For some objects, setting “stale” < “time” will make them disappear from the WinTAK/ATAK screen. If you set “stale” > “time” then when the current time = stale WinTAK greys out the object. A few seconds later (not sure exactly how many seconds) WinTAK removes the object entirely from the display.
4. “point” (the lat, lon, alt, along with the error bounds on each of these)
5. “type” (which is a string of the form x-x-x-x-x). This string can be the elements of a 2525B symbol code (if you want something rendered as a 2525B symbol), or, it could be a hierarchical list of “domain specific” values.
6. “uid” ( which is a globally unique identifier for this event. Note that uid is not an identifier for a particular message, rather, it is a unique identifier for an event. If there are further updates to this event then the uid will stay the same (but the time field will differ)).
7. “details” (this attribute allows you to extend the schema in whatever way you want by including sub-schemas).

In the WinTAK protocol if you want to delete an item then you make the “event” attribute “stale” less than the attribute “time”.

Here is a graphical representation of the CoT domain model gleaned from the FreeTAKServer code (on github at: https://github.com/FreeTAKTeam/FreeTakServer:

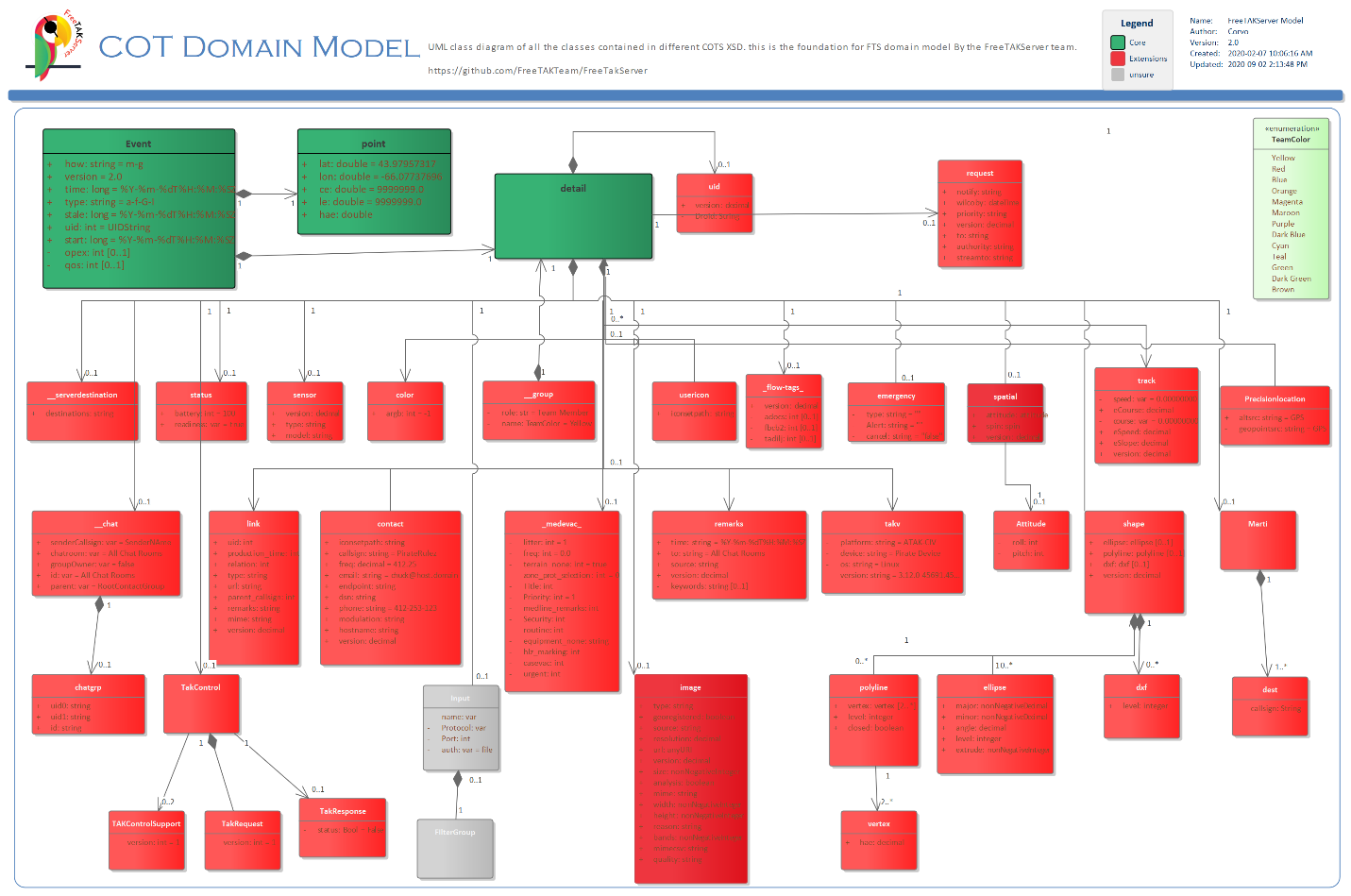


Figure CoT Domain Model

#### Data Packages

Data Packages are a way to exchange lots of tactical and graphical objects in one package. This is useful for sharing overlays, for example.

The data packages are communicated between participants (and saved to disk) as zip files.

The structure of the zip file seems to be a set of directories, with one object per directory. The name of the directory is the UID of the object. The file in the directory is an XML file which contains the CoT expression of the object (in XML). The actual filename is <UID>.cot (but it is really an XML file).

Examples of such files can be found in the local directory (for WinTAK on Windows) /Users/<user>/AppData/Roaming/WinTAK/Data Packages directory (you need to save a set of layers as a data package first!).

When a Data Package is sent to the ATAK Server it seems to be sent as a POST message.

When you post package to specific contact in ATAK, following happens[[1]](#footnote-1):

1. Datapackage is uploaded to server, recorded in database and stored in FTS directory
2. Client receives payload with URL pointing to datapackage so ATAK can download it https://github.com/FreeTAKTeam/FreeTakServer/blob/master/FreeTAKServer/README.md

## CoT code

### Source Code

1. Source code for ATAK-CIV

<https://github.com/deptofdefense/AndroidTacticalAssaultKit-CIV>

This repository contains a ton of good stuff.

1. Source code for python protobuf conversion:

<https://pypi.org/project/takprotobuf/>

It turns out that I the takprotobuf library didn’t serialize/deserialize all the objects in the ATAK CIV source code. So, I decided to do it myself.

I followed the tutorial for protobufs at: <https://developers.google.com/protocol-buffers/docs/pythontutorial>

I installed protobuf on the Ubuntu VM using the command  
sudo snap install protobuf --classic

I then went into the source code directory for the AndroidTacticalAssaultKit-CIV and copied the “takproto” directory into my Visual Studio code project

Then, in VS code project I installed the vscode-proto3 plugin: from the VS marketplace. However, I didn’t want to fart around with that so I just hand-generated the protobuf files.

I created a directory for the output files:

Mkdir takproto\_python

Then, I executed the command:

Protoc -I takproto --python\_out=takproto\_python takproto/\*.proto

This generated a bunch of python files for each of the messages in the directory takproto\_python

I then tried using the protobuf code in my python publisher. However, things didn’t go so good.

I got an error “google.protobuf.descriptor has no attribute ‘\_\_internal\_create\_key”. I googled that error and found the solution in <https://stackoverflow.com/questions/6656411>. Basically, the version of protobuf and protoc were out of sync. I issued the command:  
pip install –upgrade protobuf

Then, my protobuf and protoc were relatively in sync:  
pip show protobuf (gave me version 3.18.1)  
protoc --version (gave me 3.14.0)  
  
After I closed and restarted VS I was able to run everything! Woo-Hoo!

Documents that describe the CoT protocol:

1. Takproto/README.md – this is an excellent document that describes the tac protocol in detail.
2. <https://www.ballantyne.online/de-mystifying-the-tak-protocol/> - excellent document. Good references.
3. CoT Message Router User’s Guide 09\_4937.pdf. – pretty good reference. However, doesn’t go into many details. Describes regular expression filtering.
4. The Developers Guide To CoT a637348.pdf – Excellent document. Start understanding CoT here. Use some of the representative xsds and xmls to provide concrete examples.
5. KLV to Cursor-on-Target (CoT) Conversions.pdf – good document describing a representative CoT message
6. Reddit Tak site: https://www.reddit.com/r/ATAK/wiki/index

## WireShark Tips for working with CoT

### To filter only multicast traffic:

(eth.dst[0] & 1)

## Video Tutorials

1. WinTAK Civ Tutorial Series: https://www.youtube.com/playlist?list=PLD4gdaBHX0b7kztKgcqrXcQ69WRfwAlac
   1. WInTAK 4 0 Overview <https://www.youtube.com/watch?v=NpzrVi1R8Ws>

Very good overview of all the functionality. You need to set up the WinTAK Server and have some others online to get the full effect. But, most of the stuff described in the video works even without the server running.

* 1. WinTAK 4 0 Tools & Functions Extensive Tutorial
  2. WinTAK 4 0 TAK Chat Tutorial: <https://www.youtube.com/watch?v=DPb7HMTSiO0>: <https://www.youtube.com/watch?v=7DFvpk5l-p8>
  3. WinTAK 4 0 Maps & Favorites Tutorial: <https://www.youtube.com/watch?v=zNo76ByfbiE>
  4. WinTAK capabilities demo: <https://www.youtube.com/watch?v=-YQMo7T_NaE>
  5. WinTAK 4 0 Settings Tutorial: <https://www.youtube.com/watch?v=4IsEmWLN6pw>
  6. WinTAK 4 0 GoTo Tool Tutorial: <https://www.youtube.com/watch?v=Xk1oEAUBoCY>

1. FreeTAKServer 1.5 Setup Guide: <https://www.youtube.com/watch?v=eMh9G96-npw>  
   A good video on how to install and run the FreeTAKServer. Running on Ubuntu 20.4
2. FreeTAKServer 1.5 Overview: https://www.youtube.com/watch?v=q4BpolzIDLw
3. How to Install ATAK: https://www.youtube.com/watch?v=y6Ah5jQWu4M
4. ATAK (WinTAK) How To: Intelligence, Surveillance, and Reconnaissance – A Primer: <https://www.youtube.com/watch?v=ar3DNJaOJ1I>
5. Check FreeTAKSserver connection: https://www.youtube.com/watch?v=q-GYlTalzwQ
6. TAKGM CoT to WinTAK Demo: <https://www.youtube.com/watch?v=_TKg57Whp1s>  
   Pushing CoT messages from Unreal Engine via UDP
7. Pushing Video From Unreal Engine to WINTAK/ATAK: <https://www.youtube.com/watch?v=QldjbOEcePs>  
   Video from unreal and displayed in WINTAK/ATAK
8. TAKCAR (show Traccar devices inside of ATAK-CIV and WinTAK CIV): <https://www.youtube.com/watch?v=KvzcrZlr9bU>
9. Alert Button & Location Updates From Wintak to Unreal 4: <https://www.youtube.com/watch?v=26WXfRDGBOA>
10. TAKGM pushing CoT to WinTAK: <https://www.youtube.com/watch?v=YbfjmtkwpOc>  
    Populating an entity on WinTAK through COT protocol
11. ATAK (WinTAK) Tutorial: How to Download Better Imagery: <https://www.youtube.com/watch?v=jQJYHX24uFg>
12. ATAK (WinTAK) How to: Interior Building Maps Tutorial: <https://www.youtube.com/watch?v=_yvdxZEIQow>
13. Built in in TAKGeoCam: <https://www.youtube.com/watch?v=7EBEaEVofGM>
14. ATAK/CivTAK Data Packags: <https://www.youtube.com/watch?v=mWMsItMijHo>
15. Move, Shoot, and Communicate with ATAK (and alternatives): <https://www.youtube.com/watch?v=WXuQERL_e8M>
16. ATAK/CivTAK Overview (basic screen navigation): <https://www.youtube.com/watch?v=36Y-459PLKE>
17. CivTAK Overview, Buttons & self Marker: <https://www.youtube.com/watch?v=v1f_Tll4mJQ>
18. WinTAK 40 Overlay Manager Tutorial: <https://www.youtube.com/watch?v=aIg6It4_sJ4>
19. How to Install ATAK: <https://www.youtube.com/watch?v=y6Ah5jQWu4M>

## FreeTAKServer installation:

## <https://freetakteam.github.io/FreeTAKServer-User-Docs/Installation/PyPi/Linux/Install/>

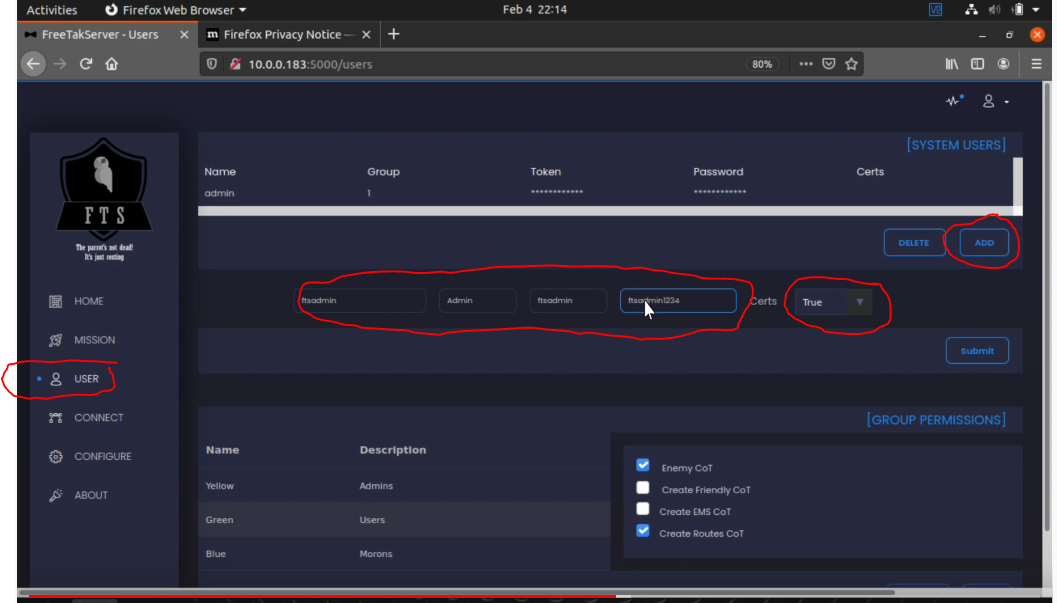
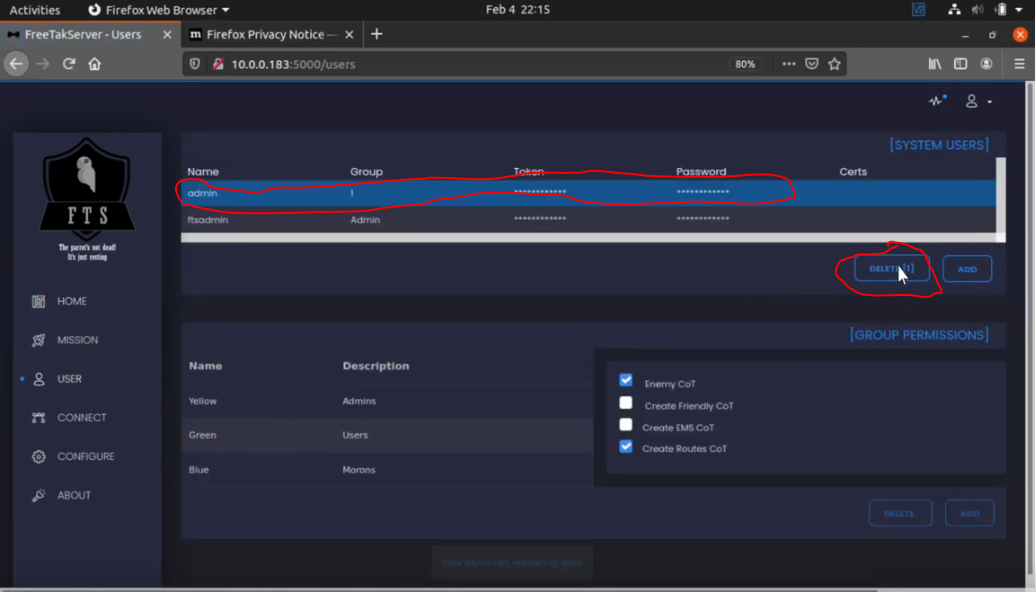
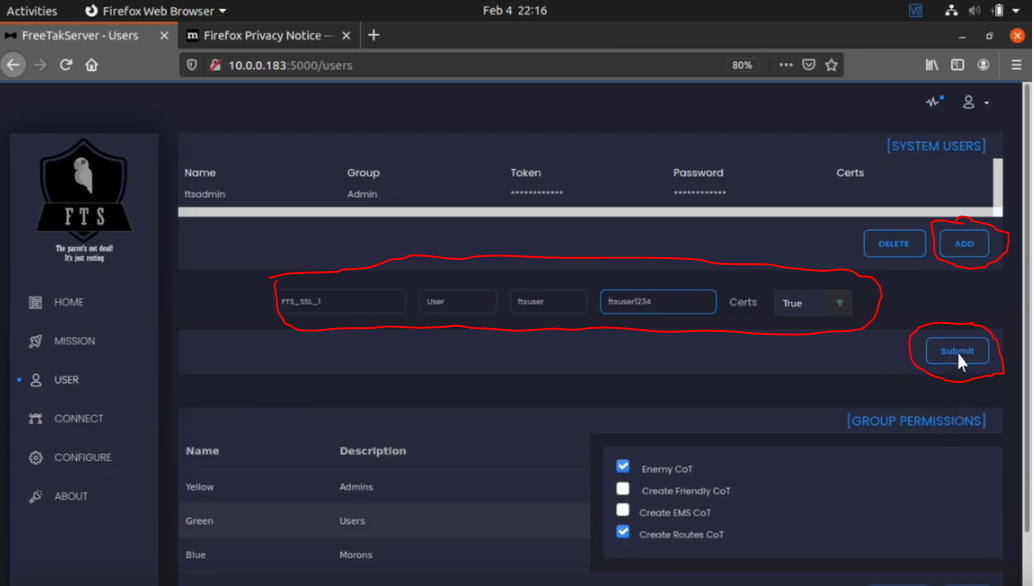
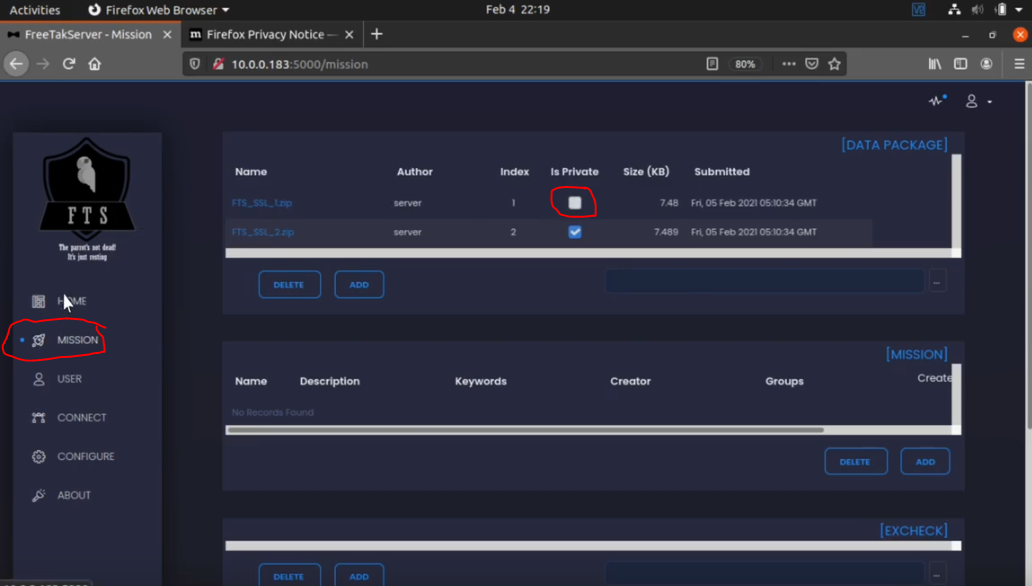
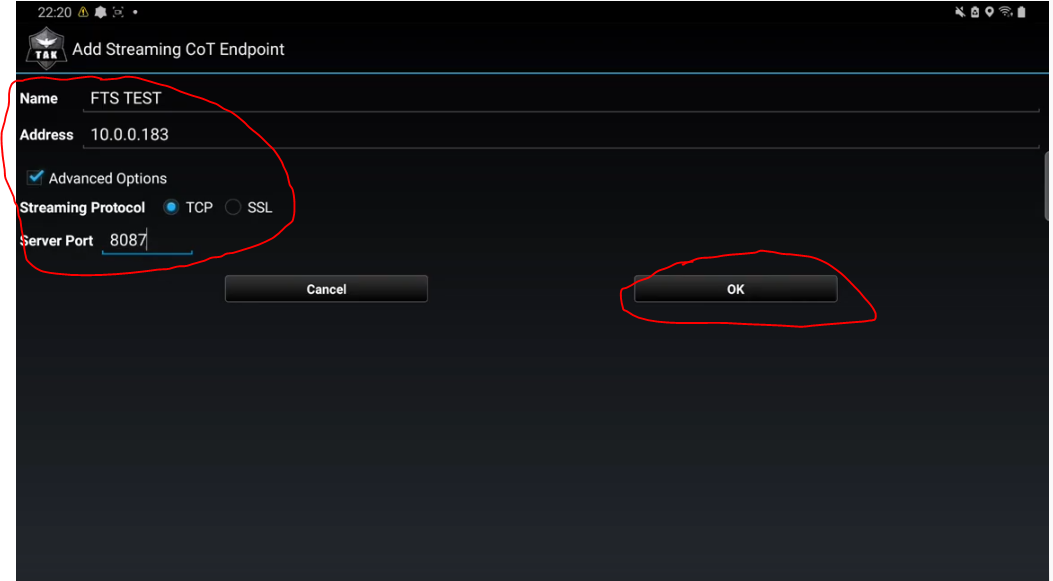
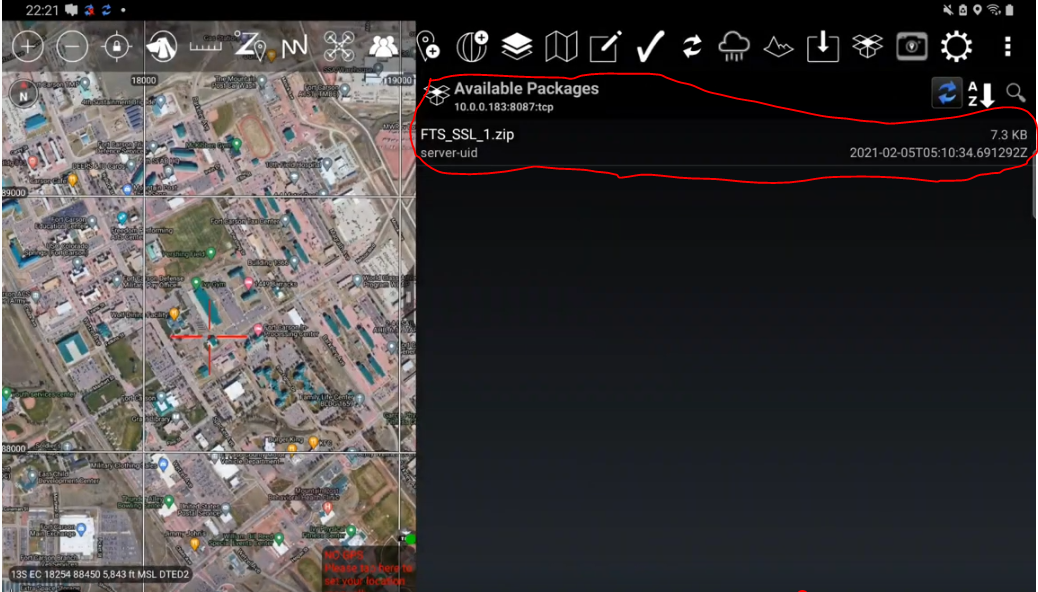
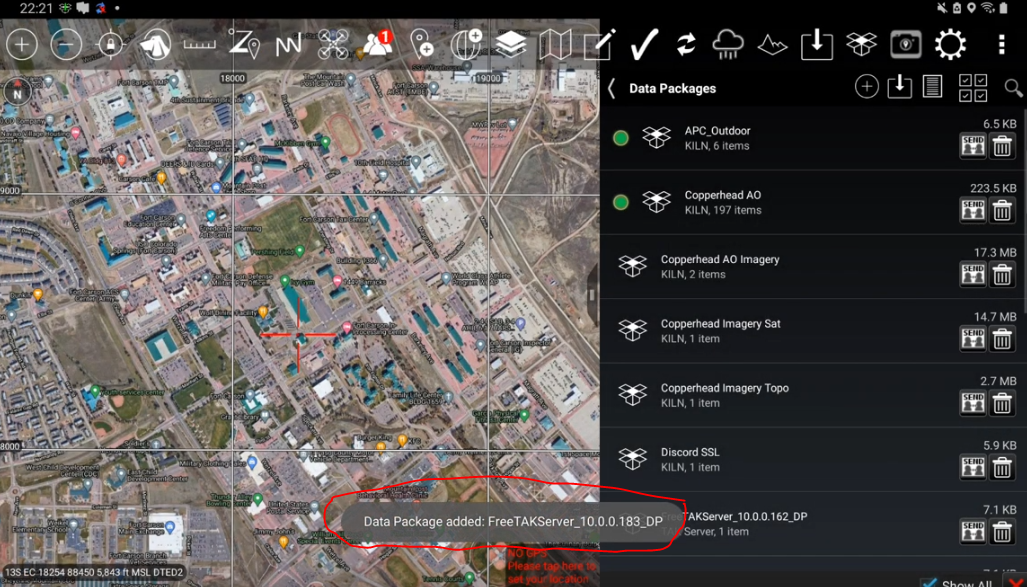
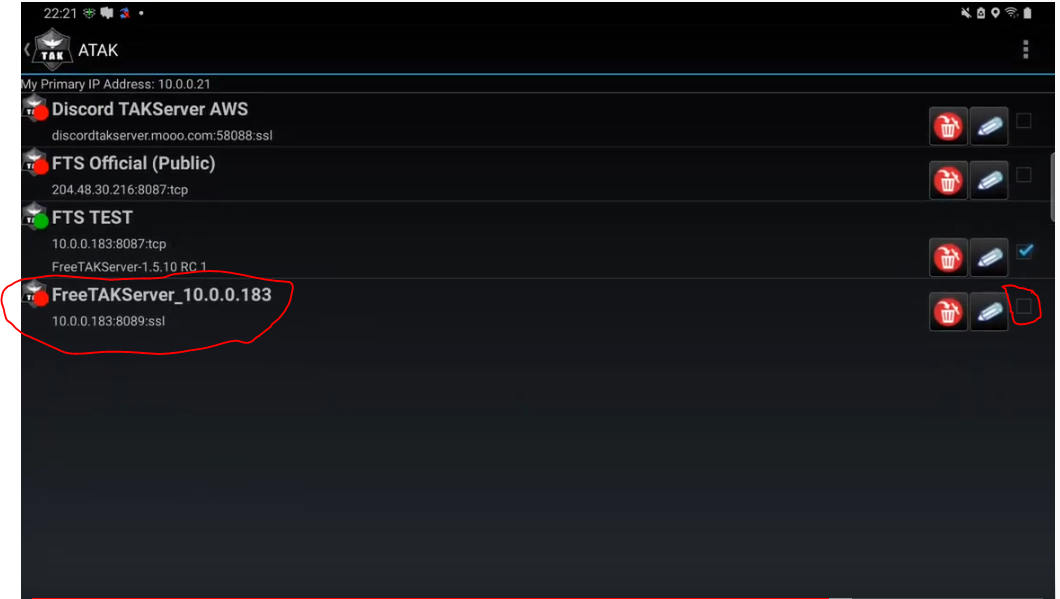
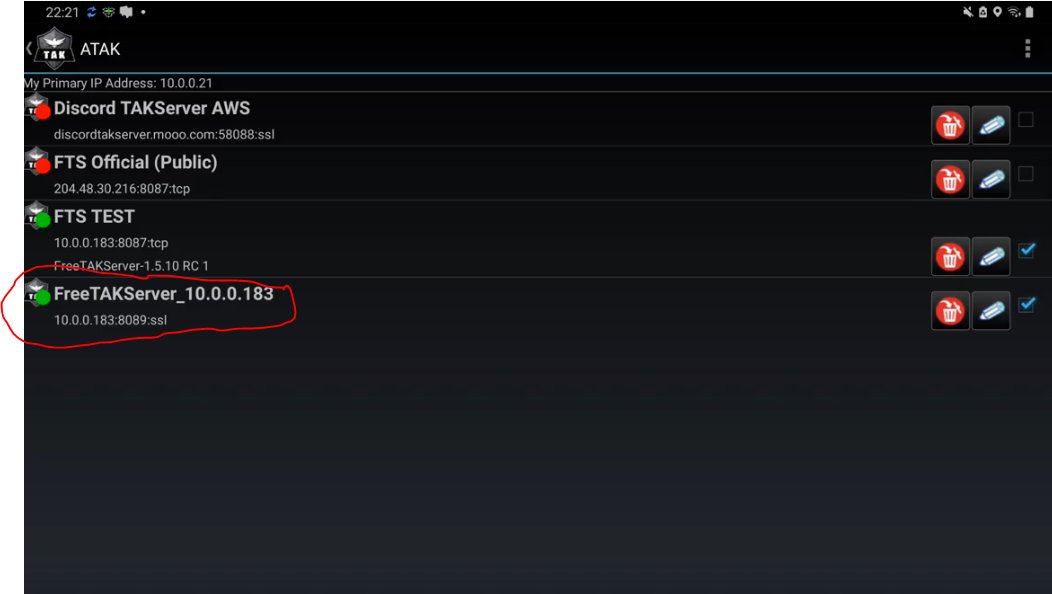
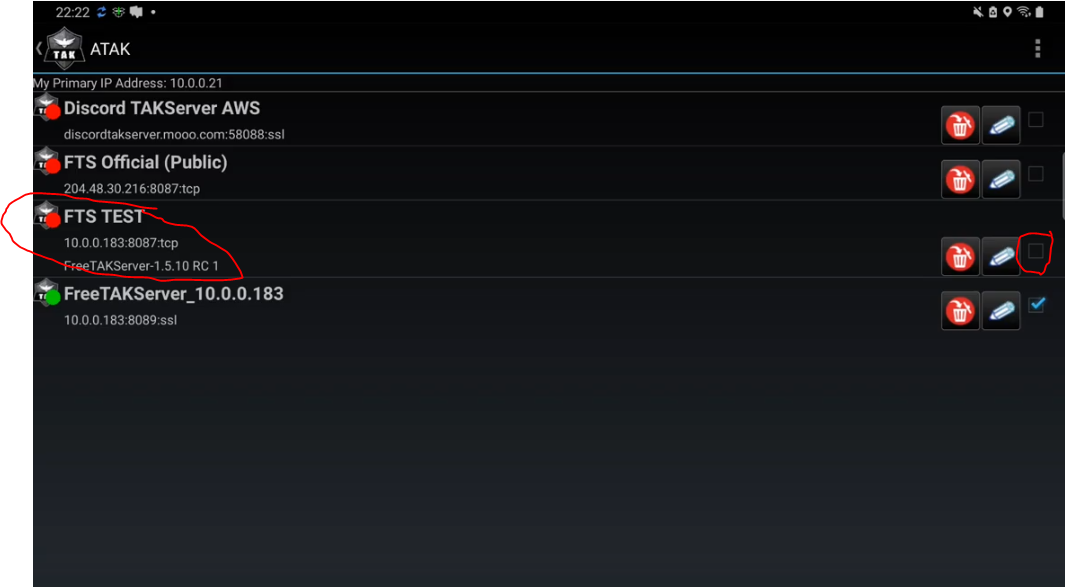
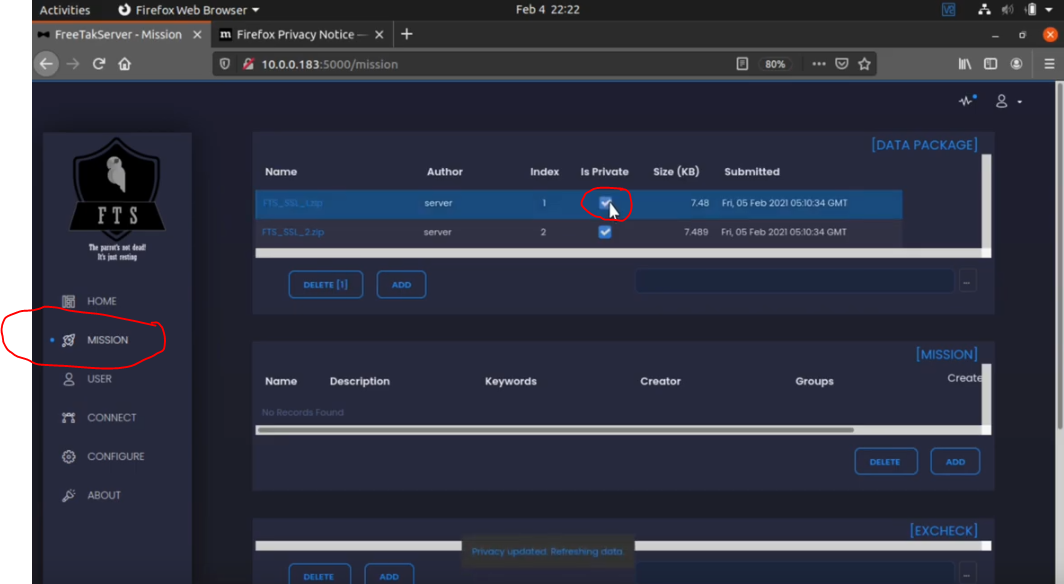
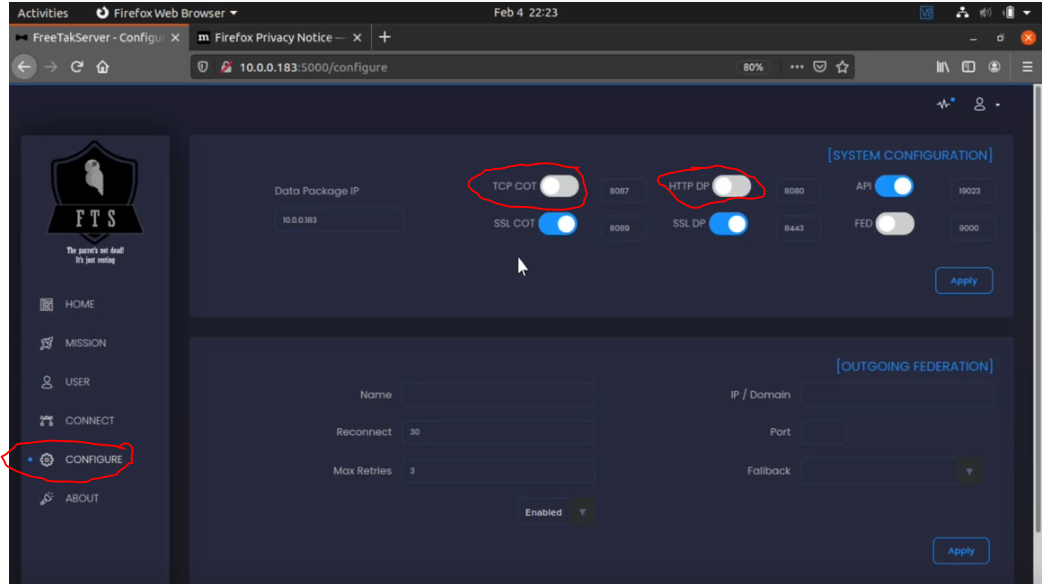
## FreeTAK Server Setup

1. FreeTAKServer 1.5 Setup Guide: <https://www.youtube.com/watch?v=eMh9G96-npw>

This is done on a linux box.

The video shows how to set up the FreeTAK Server on the local Wifi or LAN.

The FreeTAK Server is written in python. You need at least python 3.8 installed.

1. Sudo python3 -m pip install FreeTAKServer[ui]
2. Now you edit some configuration files:
   1. Find your local ip address:  
      ip addr show   
      Take a note of your IP address, you will need it in the next step
   2. Cd /usr/local/lib/python3.8/dist-packages
   3. sudo nano FreeTAKServer/controllers/configuration/MainConfig.py  
      Change the “DataPackageServiceDefaultIP = str(“0.0.0.0”) line and put in your IP address.  
      Change the “UserConnectionIP = str(“0.0.0.0”) to the IP address  
      If necessary, change the “python\_version = ‘python3.8’” and “userpath = ‘/usr/local/lib’” lines to match where you’ve installed python and which version of python you’ve installed (for example, for raspberry Pi you might only be able to get python3.7).  
      You may want to change the “federationKeyPassword = str(‘defaultpass’)” to something else (in the video the guy changes it to ‘atakatak’).and the “password = str(‘defaultpass’) to something else (in the video the guy changes it to ‘atakatak’ (which might cause some confusion because now the federationKeyPassword and the password are the same thing! But, whatever.)) Anyhow, the reason he changes it is because the “OGTAKServer” (which I presume is the TAKServer distributed by the WinTAK team) uses ‘atakatak’ as the federationKeyPassword and password and so the FreeTAK Server can then interoperate with the OGTAKServer.  
      You may also change the “ConnectionMessage = f’….’” line to customize your own connection message
   4. sudo nano FreeTAKServer-UI/config.py  
      Change the “APPIP = ‘127.0.0.1” line to ‘0.0.0.0’  
      Change the “IP=’127.0.0.1’” line to the IP address of your linux machine  
      If you are using raspberry pi OS then you may also need to change the python version in the strings for the certpath
3. To Start the Server  
   sudo python3 -m FreeTAKServer.controllers.services.FTS
4. To start the UI  
   sudo FLASK\_APP=/usr/local/lib/python3.8/dist-packages/FreeTAKServer-UI/run.py python3 /usr/local/lib/python3.8/dist-packages/FreeTAKServer-UI/run.py
5. To test if the server is running open a web browser window to the IP address of the machine running the FreeTAKServer, port 5000  
   10.0.0.183:5000 (in the example in the video)  
   Username=admin, Password=password
6. When you are in the UI then click on “USER” in the left hand navigation bar, and then click “Add”  
   Then, make up a username (ftsadmin), a group (admin), a token (ftsadmin1234) and a password (ftsadmin1234). Set the Certs to “false” (for some reason we don’t’ want to have certificates for the admin). (I also had to set a “Token” field (which wasn’t in the video). I set token=ftsadmin1234)   
   Now, logout.
7. Log back in. When you log back in you use the ftsadmin user account, with the password ftsadmin1234
8. Now, delete the “admin” user, which only leaves the ftsadmin account available:  
   
9. And now add a user account that will allow clients to connect using SSL  
   FTS\_SSL\_1  
   user  
   ftsuser  
   ftsuser1234  
   Certs=”True”  
   
10. Now, if clients are going to connect to the server and download the SLL we have to make sure that the Mission is not private. To do this you select “MISSION” from the left hand navigation bar, then unselect the “Is Private” control:  
    
11. Now we go to ATAK (or WinTAK) and go to:
    1. Settings
    2. TAKServers
    3. Add
    4. Name = “FTS TEST” (or whatever you like)
    5. Address=10.0.0.184 (the IP address of the machine running the TAKServer)
    6. Advanced Options
       1. TCP
       2. Server Port 8087
    7. 
12. You should now be connected to the server!
13. Now, in ATAK/WinTAK we go to “Data Packages” and select the server that we created, and we will see the SSL data package that we created earlier on our server. Download that data package:  
      
    
14. In ATAK/WinTAK go into “Settings”, “TAKServers”, and you will see a TAKServer with the IP address of the machine (10.0.0.183), and click on the “enable” check box:  
      
    and you will get:  
    
15. Now, uncheck the FTS TEST connection (since you won’t want to use non-SSL in production):  
    
16. Once all the clients are done downloading you can go back to the FreeTAKServer UI and select “Mission” and check the “Is Private” on the FTS\_SSL\_1 server (so that no-one else can download the certificates):  
    
17. Now, to further secure the server shut down the HTTP and TCP connections. In FreeTAKServer UI go into “Configure” and de-select the TCP COT and HTTP DP:  
    

## FreeTAKServer overview:

<https://www.youtube.com/watch?v=q4BpolzIDLw>

## Working with the Ubuntu VM

I created a Linux VM in VirtualBox and installed the Ubuntu 20.04 iso. 2GB memory, 100GB disk (dynamically allocated)

I then installed wireshark following the instructions at: https://linuxhint.com/install\_wireshark\_ubuntu/

Sudo apt update

Sudo apt install wireshark

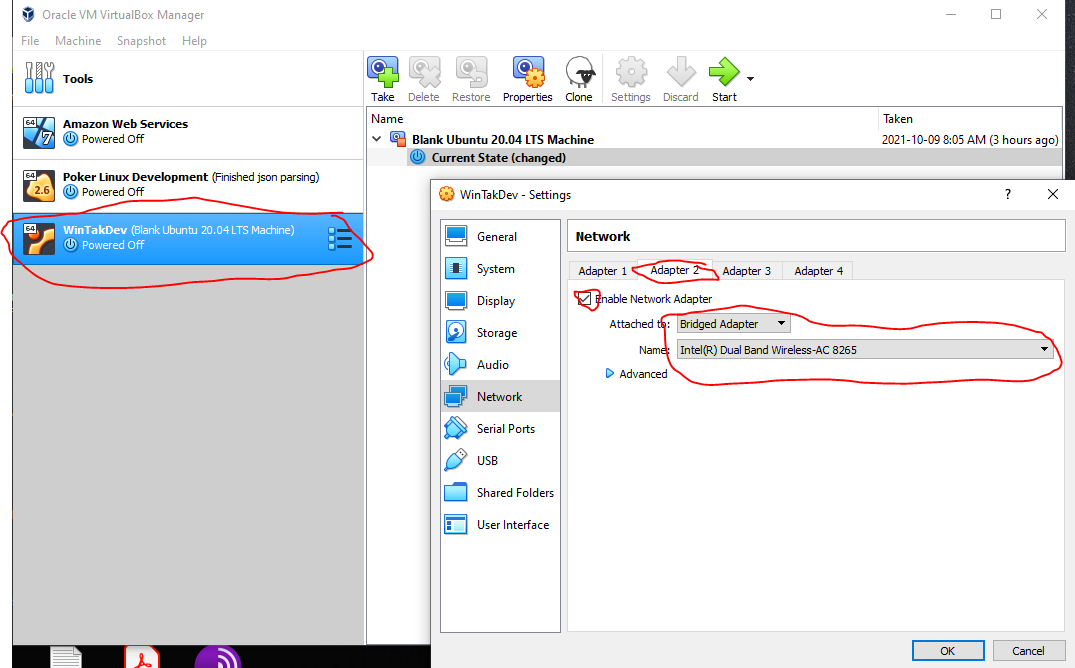
I then restarted the VM

Then, I ran wireshark from a command window:

Sudo wireshark

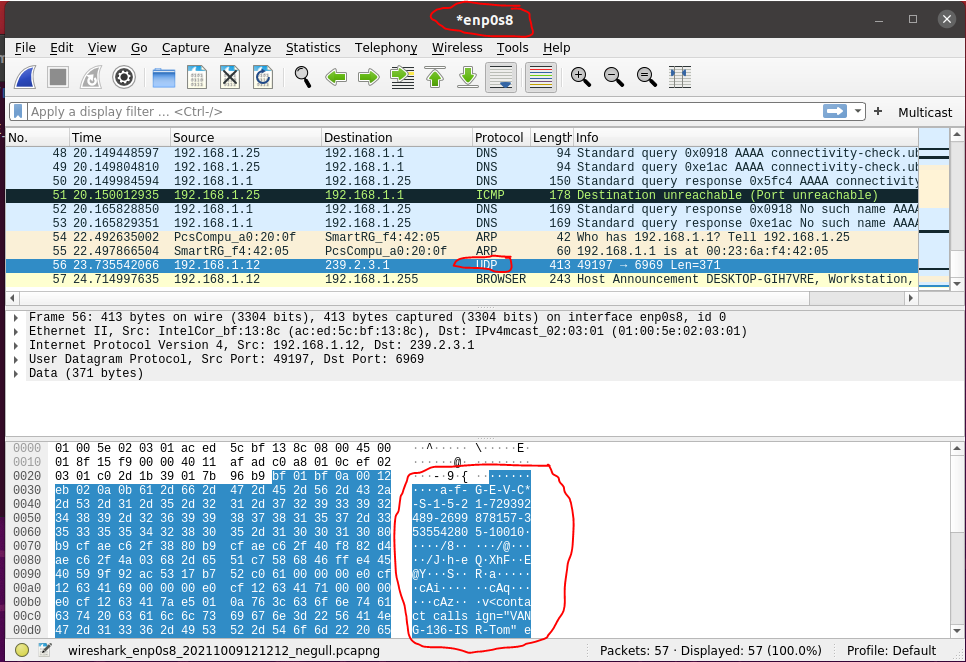
When I tried doing a capture on the enp0s3 interface I did not see any UDP packets.

I had to shut down the VM, and then allow bridging of the VM’s network to the Wifi network of my host.



Once I did this then another network interface appeared in my VirtualBox VM (which corresponded to the wifi network):

And, in the VM I was now able to see the UDP packets being sent out by winTAK on my wifi (as shown below).



Now, it turns out that python3.8 is already installed by default in Ubuntu 20.04, so I’m good to go there.

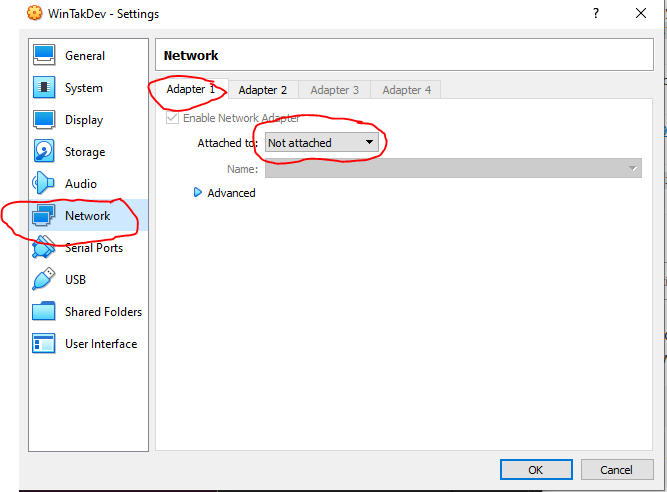
Next, I installed vscode. I went to the vscode web page and it had instructions

Sudo apt install code

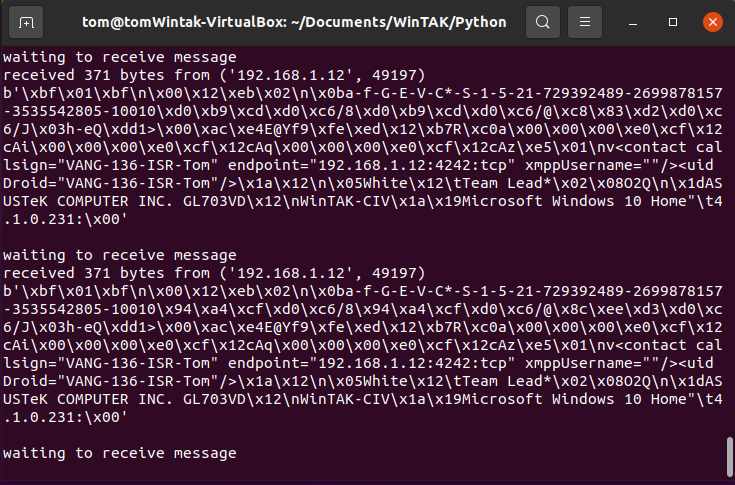
I then wrote two python programs (a multicast publisher and subscriber) following the examples given at:  
https://pymotw.com/2/socket/multicast.html

However, even though the python programs were able to communicate amongst themselves on the VM using multicast, they were not able to hear the messages coming from the host (which was the wifi network connected to client interface ‘enp0s8’). It seems that the python programs were somehow defaulting to the NAT interface on the client.

To get the UDP packets I had to disable the NAT interface on the client. So, that’s what I did. On the client I went into “Machine->Settings” and then I set the “Attached to:” option to “Not attached”. After doing this it took a couple of minutes for the client to rebuild its routing, but once it did that I was able to use a web browser inside the client (which showed that it had connectivity to the host, and I was also able to see the multicast packets coming from the host! Woo-Hoo!).



Here’s the output of the multicast packets that came from the host:



The structure of this message is described in the documents:  
<https://www.ballantyne.online/de-mystifying-the-tak-protocol/>

From that document, the message starts out with:

<magic byte><tak protocol version><magic byte><tak protocol message>

The magic bytes are 0xbf in hex, and the protocol version can currently be either 00 for XML payload or 01 for Protobuf payload.

In this case captured in the message above the <magic bye> is \xbf (0xbf) and the protocol version is \x01 (0x01). This means that the message is encoded as a protobuf message.

So, I need to convert to/from protobuf.

For this used the library at:

https://pypi.org/project/takprotobuf/

And installed it in the Ubuntu VM using:

Pip install takprotobuf

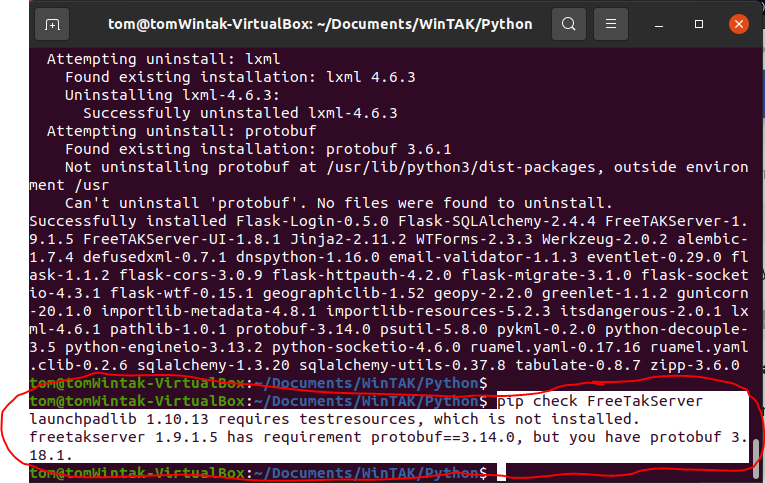
Okay, so that all went awry. After a while I was unable to connect from my VM to my host machine via wifi. Wireshark could see the wifi, but my python scripts could not. It turns out that I had to restart VirtualBox to solve the problem. For some reason VirtualBox messed up (not the VM itself). Anyhow, after I restarted VirtualBox and restarted the VM then I was able to run the python subscriber again and get messages.

## Installing FreeTakServer

I followed the instructions for installing the FreeTakServer at:

<https://freetakteam.github.io/FreeTAKServer-User-Docs/Installation/PyPi/Linux/Install/>

When I got to the part about checking the install using pip I got the message:



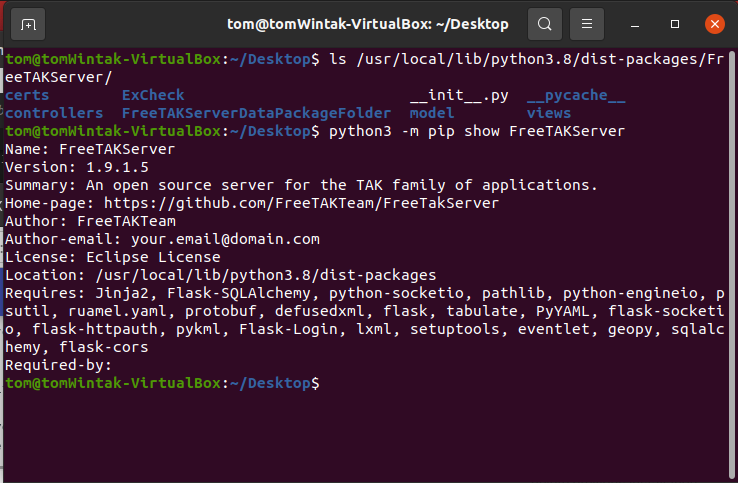
I continued on anyways.

I put the server config file in /opt/FTSConfig.yaml

The log files for the FreeTAKServer are in /usr/local/lib/python3.8/dist-packages/FreeTAKServer/Logs

I got information about the freetak server by running the command

Python3 -m pip show FreeTAKServer



I then started the freeTAKServer by running the command:

Sudo python3 -m FreeTAKServer.controllers.services.FTS -DataPackageIP 0.0.0.0 -AutoStart True

This seemed to work okay.

I also created a FreeTAKServer .service file (as in the documentation) and I reloaded system to start the service

Alternatively, I started the FreeTAKServer using the command line:

Sudo python3 -m FreeTAKServer.controllers.services.FTS

I started the FreeTAKServer GUI backend service using the command line:

Sudo FLASK\_APP=/usr/local/lib/python3.8/dist-packages/FreeTAKServer-UI/ run.py python3 /usr/local/lib/python3.8/dist-packages/FreeTAKServer-UI/run.py

1. This documentation is from https://github.com/FreeTAKTeam/FreeTakServer/blob/master/FreeTAKServer/README.md [↑](#footnote-ref-1)