VHF management Status and Plans

A.Formica, H. Louvin, JP Le Fevre, E. Trigui

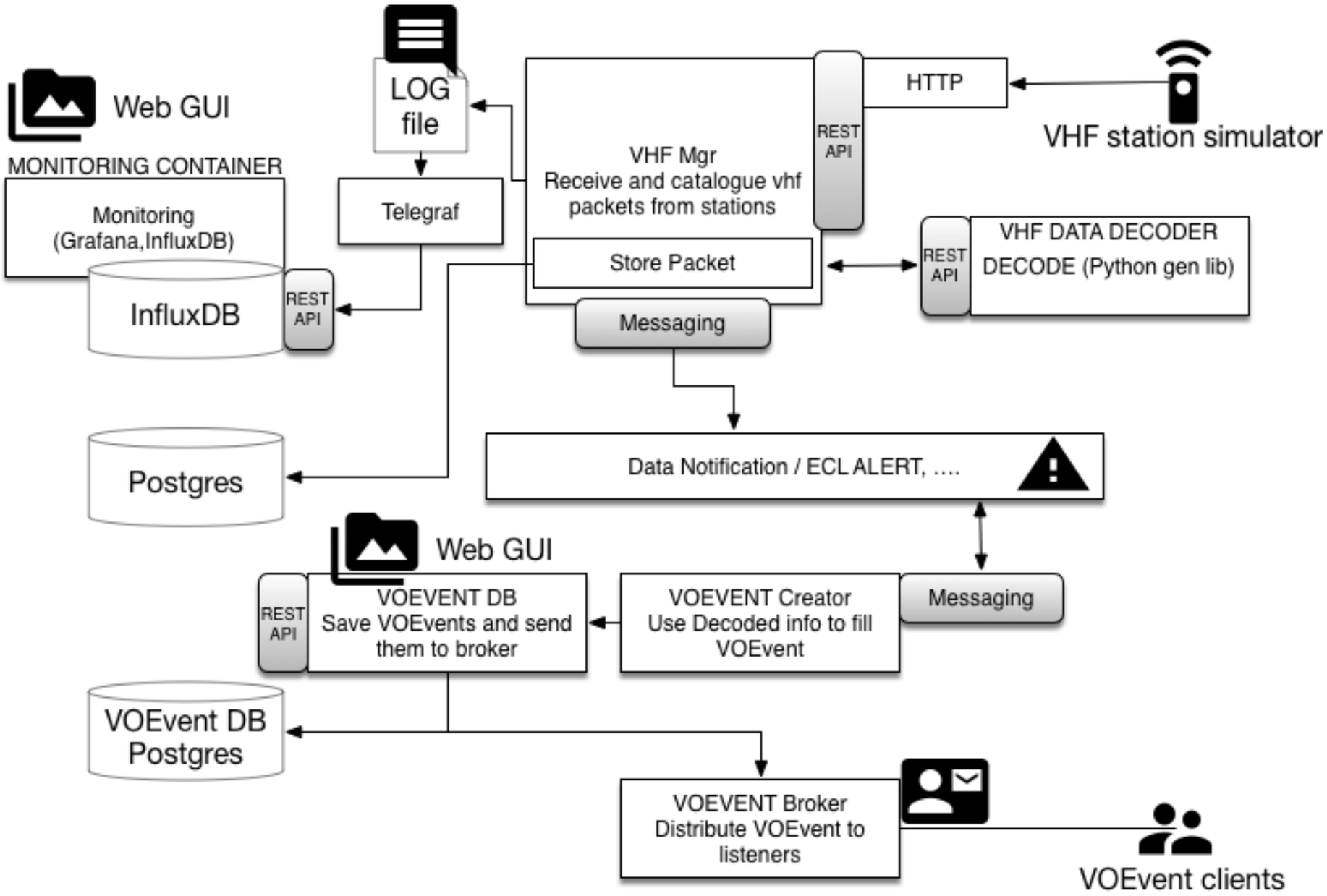
Outline

- What do we have today ?
  - The DC0 prototype example…
  - On going developments : the beginning step for pipelines running on VHF data
- Plans
  - Interacting with VHF manager system from clients via REST (HTTP) : defining the methods for data retrieval

DC0 : VHF services

- VHF simulator
  - Simulate the packet generation using an input file from Marie Claire representing an orbit sequence
    - Stations which are “visible” send packets to FSC service
    - Initial time of the orbit is configurable in order to generate different packets (useful because we want to avoid duplicates)
- VHF manager (REST + messaging)
  - Decoding of station header, frame headers, filtering of duplicates etc
  - Use a dedicated service (**decoder**) to decode the internal packet content (APID specific)
- VHF decoder (REST)
  - Decode the packets internal content using generated python code from Marie Claire XML database describing the data formats
- VOEvent Creator
  - Listen for Alerts from VHF manager and create VOEvents
  - Use a dedicated services for VOEvent storage and distribution (**broker**, **voevent-db**)
- VOEvent DB and Broker (REST)
  - Storage and distribution of VOEvents
  - Provide a REST API for retrieval of generated events.

DC0 VHF services architecture



DC0 processing

- VHF sequence:
  - Satellite simulated data send from “vhf-ground-stations”
  - Data stored, decoded, ….
  - DataNotification on: Eclair + Grm alerts
    - VOEvent is generated
- Remarks:
  - Simulated data correspond to a satellite orbit, and take into account VHF stations in visibility, times, APIDs,…
  - Started LC for ECL recently….

On going developments

- Aggregate VHF packets
  - Input needed: number of expected packets for an OBS-ID
    - Information already available in the VHF database
  - Scheduled tasks
    - Verify number of packets per APID and OBSID on regular basis: once reached the “desired” (expected - n(timeout)) then trigger a processing (send DataNotification via NATs)
  - DataNotification on…(the light curve example):
    - LC number of packets received ***close to nominal*** (22 for priority / 42 for normal packets)
    - Provide an URL to retrieve the packets from VHF database

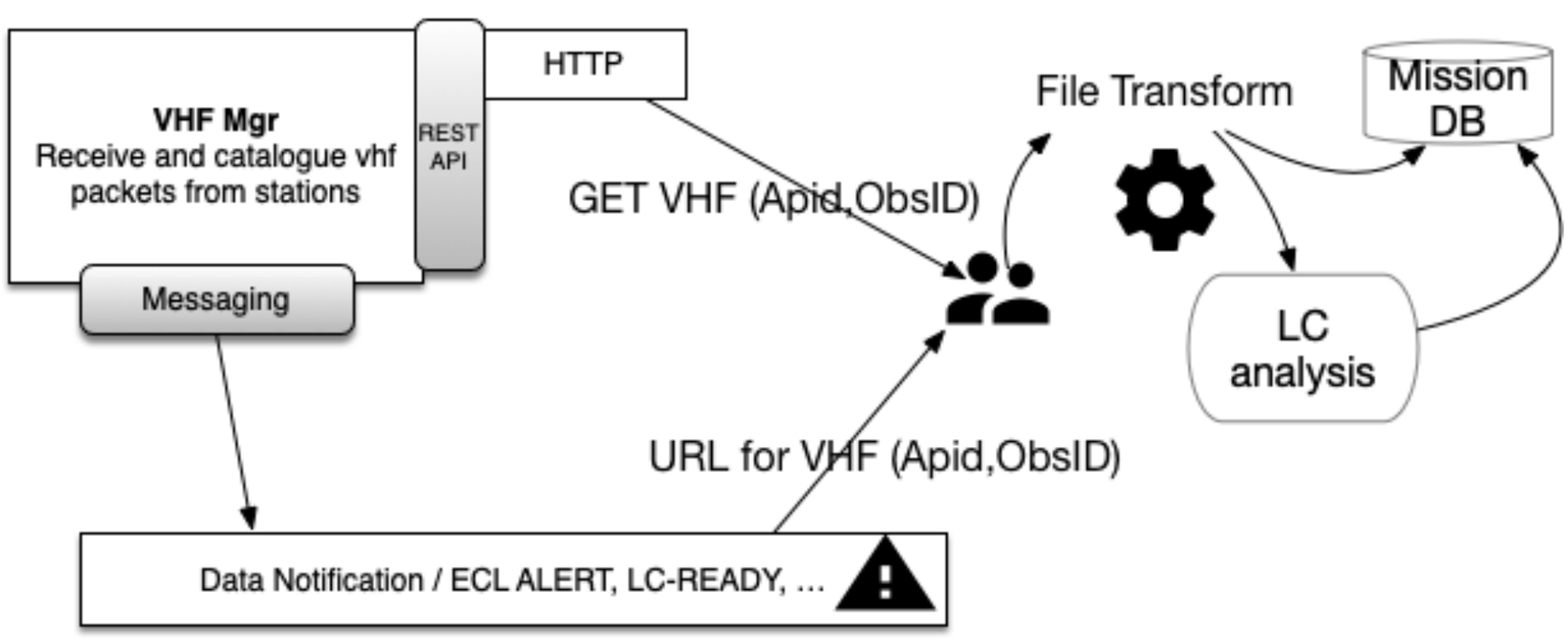
Schedulers details

- Vhf Manager can trigger schedulers (a bit like *cron task*)
  - A ***scheduler*** is a function executed at some rate (fixed rate or delay, use cron expression,…)
    - @Scheduled(cron = "0 15 10 15 * ?")
  - In this function we can read back some data and process them…several use cases for this kind of processing

- The ObsId-Apid counter scheduler
  - Simplified logic (still under development): check the packets “recently” inserted and count the number of packets by APID and ObsID.
  - Save/Update the counters at each execution
  - Use the counters to trigger further actions: when a “list” of APID packets is completed (n_pkts = n_expected_pkts) then send a notification to retrieve this list (TBD)

VHF pipelines

- How do pipelines interact with VHF mgr ?
  - Get notified that something new is there
  - Get VHF data
  - Transform data into Pipeline Inputs (FITs, …)

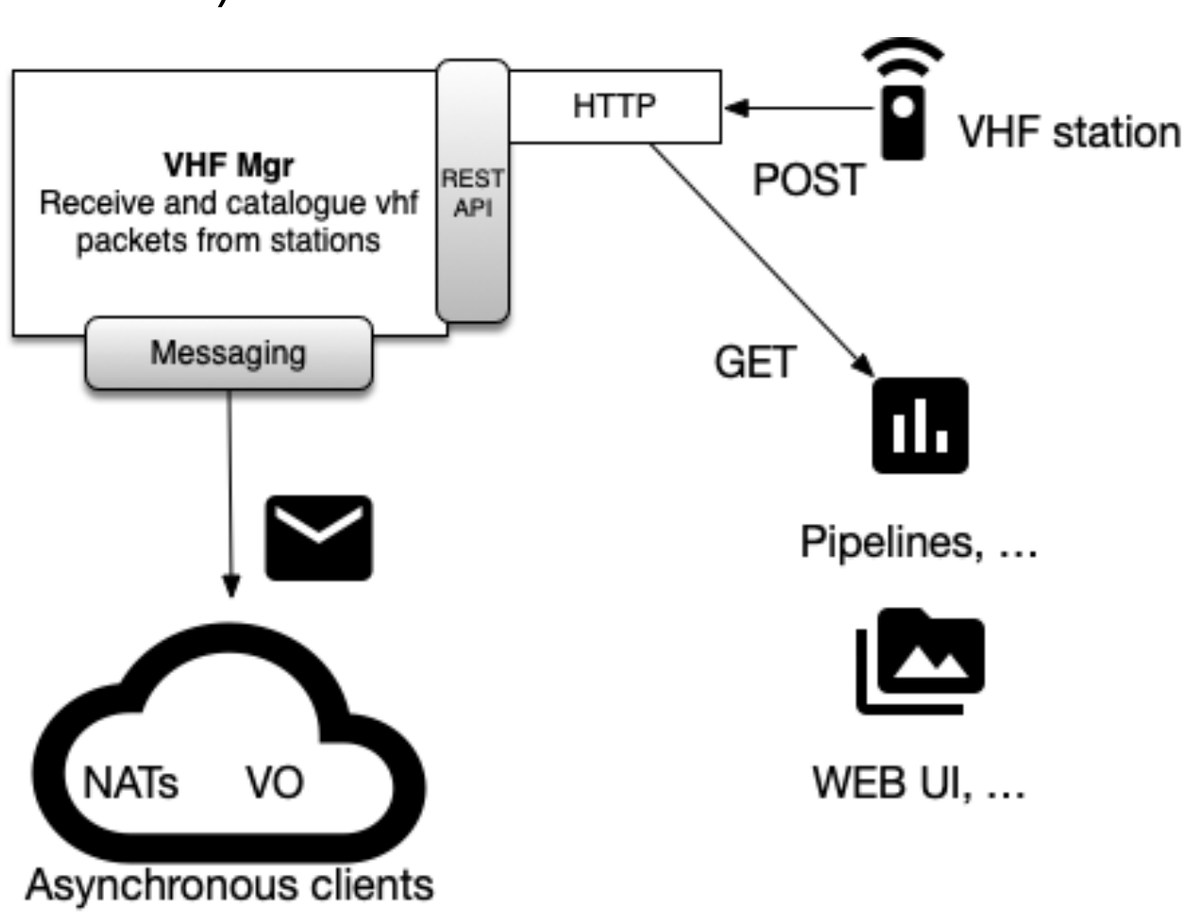


Integrating a pipeline….

- This is a sort of draft recipe
  - More to give an idea than to lead to an implementation
  - Real steps should be more clear later on
- Define Inputs (APIDs) and formats
  - All “APID” inputs should be well defined for a given pipeline, and data formats as well
  - In some cases the pipeline could directly use the decoded JSON string, in some other it will need some dedicated processing to produce the correct “input”
- Define the rules for launching the pipeline
  - Each pipeline will need a set of rules that can trigger the processing
    - I need “this and that” packet types, then optionally these others….
- Define the API to start the pipeline, check status, ….
  - Inside FSC we started a set of rules on “standard” HTTP methods and BODY formats for the requests, in order to use “as common as possible” solutions
- Define the “clients” : messaging and Mission DB (and eventually VHF Mgr)
  - To retrieve and store data we may have several possibilities, a pipeline needs to be “ready” to integrate client code for accessing different services inside FSC

VHF manager clients

- All clients use the same HTTP API
  - VHF stations need to be authenticated (x509 in place)
  - Other clients only retrieve data ….(authentication policy yet to be defined)



REST API

Documentation for API Endpoints

All URIs are relative to *http://localhost:8080/api/v1/*

Class	Method	HTTP request	Description
ApidsApi	**create_packet_apid**	**POST** /apids	Saves PacketApid
ApidsApi	**find_packet_apids**	**GET** /apids/{id}	Finds PacketApid by id
ApidsApi	**list_packet_apids**	**GET** /apids	Finds a packet APID list.
StationsApi	**create_vhf_ground_stations**	**POST** /stations	Saves VhfGroundStation
StationsApi	**find_vhf_ground_stations**	**GET** /stations/{id}	Finds VhfGroundStationDto by id
StationsApi	**list_vhf_ground_stations**	**GET** /stations	Finds a VhfGroundStationDtos list.
VhfApi	**decode_packet**	**GET** /vhf/packet/decode	Decode a VHF packet.
VhfApi	**find_vhf_transfer_frames**	**GET** /vhf/{frame}	Finds vhf transfer frame by ID.
VhfApi	**get_vhf_packets**	**GET** /vhf/packet/{hash}	Finds vhf packet by HASH.
VhfApi	**list_vhf_counters**	**GET** /vhf/count	Finds a list of summary data (ObsIdApidCountDto).
VhfApi	**list_vhf_raw_packets**	**GET** /vhf/rawpackets	Finds a RawPacketDtos lists.
VhfApi	**list_vhf_station_status**	**GET** /vhf/stationstatus	Finds a VhfStationStatusDtos lists.
VhfApi	**list_vhf_transfer_frames**	**GET** /vhf	Finds a VhfTransferFrameDtos lists.
VhfApi	**save_packet_from_stream**	**POST** /vhf/upload	Saves VhfTransferFrame from binary stream
VhfApi	**search_vhf_packets**	**GET** /vhf/packet/search	Finds vhf packet by query.
VhfApi	**search_vhf_transfer_frames**	**GET** /vhf/search	Finds a VhfTransferFrameDtos lists.

Vhf station

Data model

The base model from which we can retrieve everything else

VhfTransferFrameDto

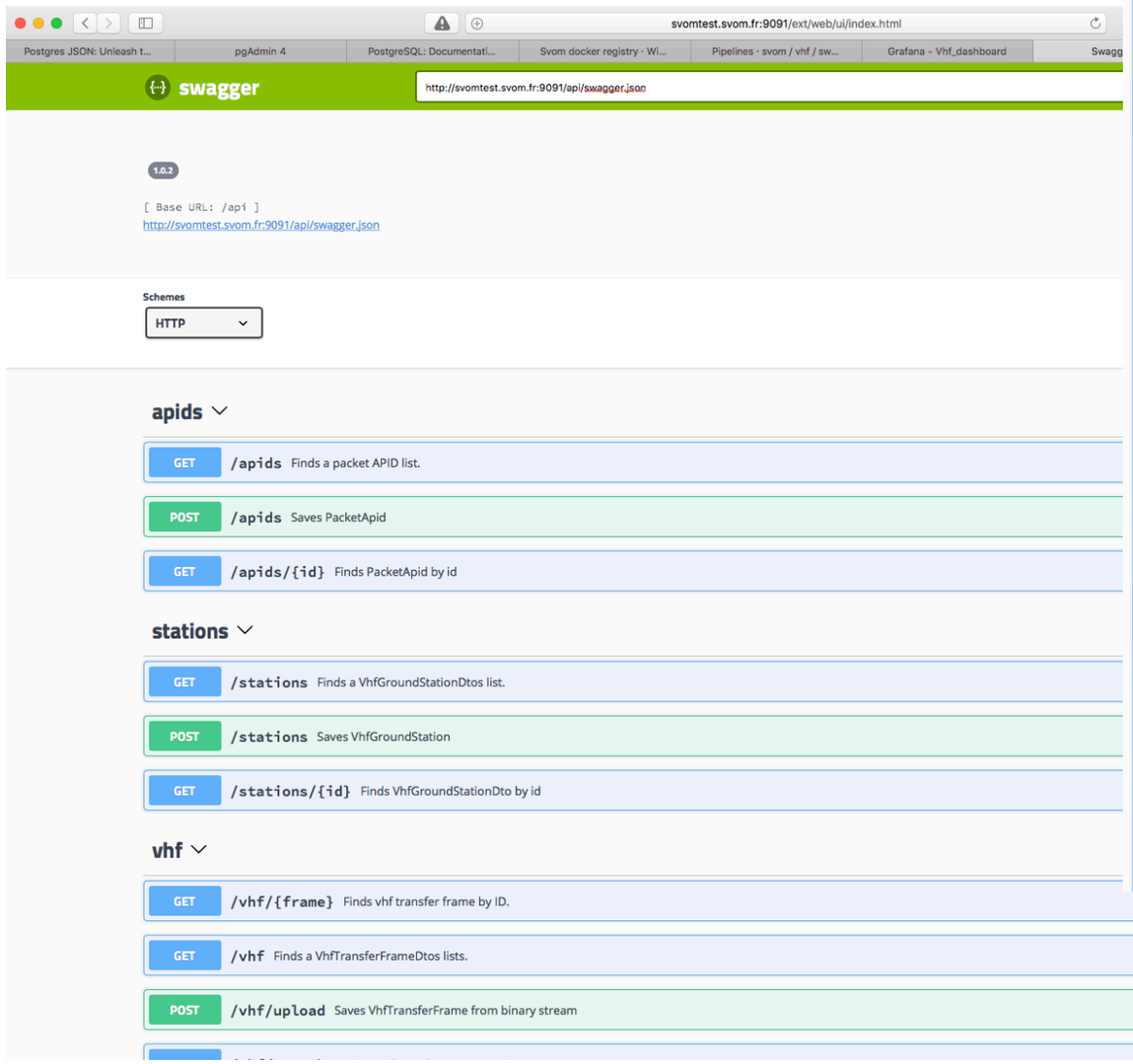
Properties

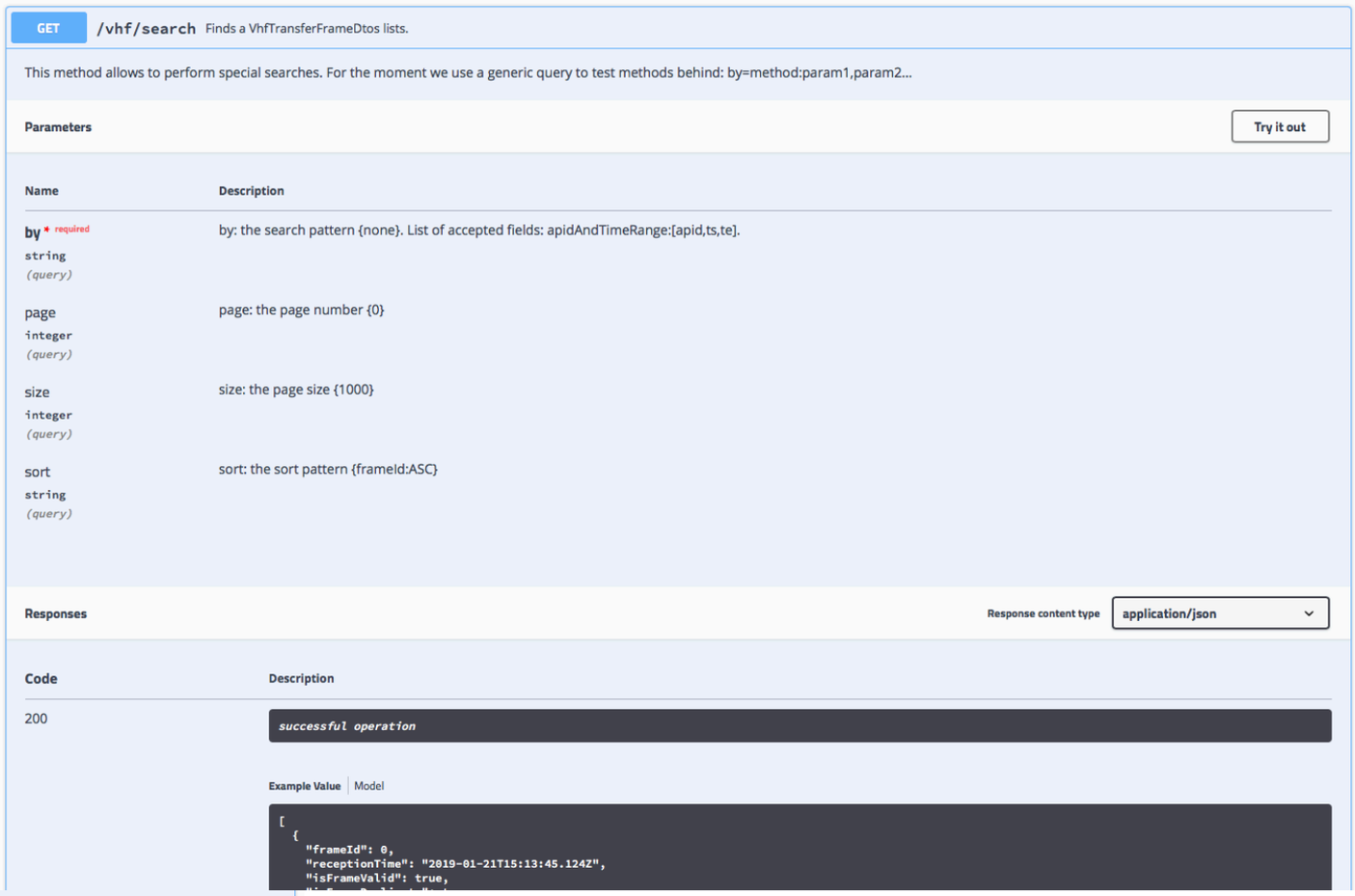
Name	Type	Description	Notes
frame_id	**int**	| [optional]	
reception_time	**int**	| [optional]	
is_frame_valid	**bool**	| [optional] [default to False]	
is_frame_duplicate	**bool**	| [optional] [default to False]	
dup_frame_id	**int**	| [optional]	
apid	**PacketApidDto**	| [optional]	
station	**VhfGroundStationDto**	| [optional]	
frame_header	**FrameHeaderPacketDto**	| [optional]	
ccsds	**CcsdsPacketDto**	| [optional]	
header	**PrimaryHeaderPacketDto**	| [optional]	
station_status	**VhfStationStatusDto**	| [optional]	
packet	**VhfBinaryPacketDto**	| [optional]	
packet_json	**VhfDecodedPacketDto**	| [optional]	
binary_hash	**str**	| [optional]	

[Back to Model list] [Back to API list] [Back to README]

Service documentation

- Access to swagger UI
  - Some examples below: **endpoints** and models





Service documentation

- Access to swagger UI
  - Some examples below: endpoints and **models**

VhfBinaryPacketDto { *description: data container for binary format, inherit from generic vhfbasepacket*

pktformat*	string	
pkttype*	string	
hashId*	string	
insertionTime	integer($int64)	
uri	string	
pktsize	integer($int64)	
packet	string	A string representation of the packet content (can be JSON or base64 string)
binarypacket	[string($byte)]	The byte array representation of input packet, this parameter is optional

}

VhfDecodedPacketDto { *description: data container for json format, inherit from generic vhfbasepacket*

pktformat*	string	
pkttype*	string	
hashId*	string	
insertionTime	integer($int64)	
uri	string	
pktsize	integer($int64)	
packet	string	A string representation of the packet content (can be JSON or … string)

}

VhfStationStatusDto {

sstatusId	integer($int64)
idStation	integer($int32)
channel	integer($int32)
padding1	integer($int32)
stationTime	integer($int32)
corrPower	integer($int32)
padding2	integer($int32)
doppler	number($float)
dopDrift	number($float)
srerror	number($float)
reedSolomonCorr	integer($int32)
ckSum	integer($int32)
padding3	integer($int32)

}

Retrieve packets info and counts

svomtest.svom.fr:9091/mgmt/h | Grafana - Vhf_dashboard | localh

Not Secure | svomtest.svom.eu:5591/api/vhf?size=10

Apps | Bookmarks | Add Site to Comm... | iCloud | iGoogle | Blogger: Dashboa


```
[
  - {
      frameId: 1,
      receptionTime: 1548108410829,
      isFrameValid: true,
      isFrameDuplicate: false,
      dupFrameId: null,
    - apid: {
        apid: 587,
        packetName: "ECLRECURR2",
        packetDescription: "none",
        className: "none",
        instrument: "none",
        category: "none",
        expectedPackets: null
      },
    - station: {
        stationId: 7,
        stationName: "auto-added",
        idstation: "0x00000007",
        location: "FSC",
        loc: "Unknown",
        description: "Unknown vhf",
        macaddress: "0.0.0.0"
      },
    - frameHeader: {
        fId: 1,
        tframeVersion: 0,
        spaceCraftId: 391,
        vcId: 3,
        ocFlag: 0,
        mcfCount: 0,
        vcfCount: 0,
        dfStatus: 6144
      },
    - ccsds: {
        ccsdsId: 1,
        ccsdsVersionNum: 0,
        ccsdsType: 0,
        ccsdsSecHeadFlag: 0,
        ccsdsApid: 587,
        ccsdsGFlag: 3,
        ccsdsCounter: 1,
        ccsdsPlength: 87,
        ccsdsCrc: 60531
      },
```


svomtest.svom.fr:9091/mgmt/h | Grafana - Vhf_dashboard | localhost:8080/coolrapi/s

Not Secure | svomtest.svom.eu:5591/api/vhf/count?from=0&apid=all

Apps | Bookmarks | Add Site to Comm... | iCloud | iGoogle | Blogger: Dashboard | Google Ma

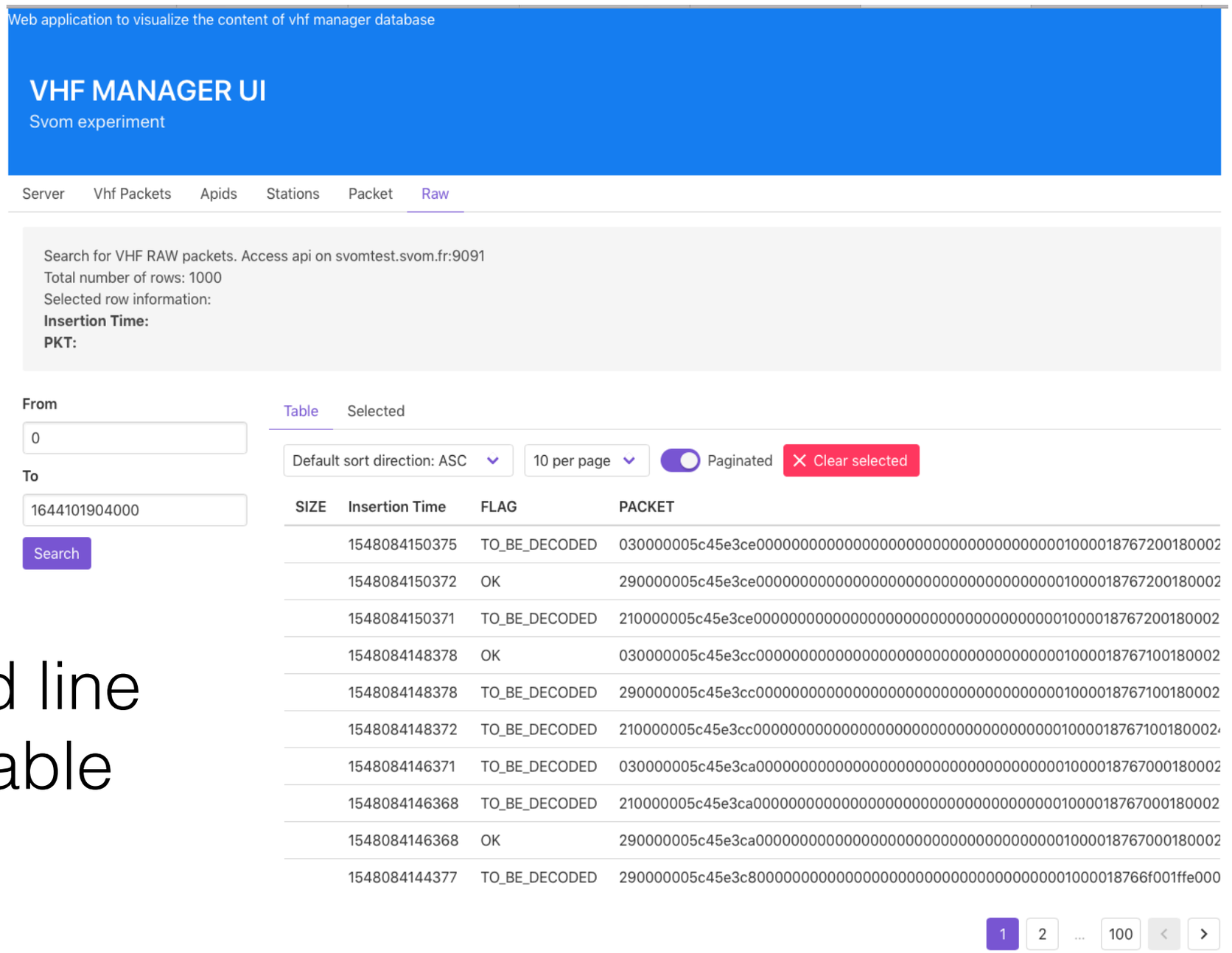

```
[
  - {
      apid: 546,
      packetName: "GRMLCURHIP",
      packetObsid: 6,
      status: null,
      countedPackets: 3,
      expectedPackets: 0,
      insertionTime: null,
      updateTime: null
    },
  - {
      apid: 587,
      packetName: "ECLRECURR2",
      packetObsid: 1,
      status: null,
      countedPackets: 186,
      expectedPackets: 0,
      insertionTime: null,
      updateTime: null
    },
  - {
      apid: 576,
      packetName: "ECLALERTL1",
      packetObsid: 6,
      status: null,
      countedPackets: 4,
      expectedPackets: 0,
      insertionTime: null,
      updateTime: null
    },
  - {
      apid: 521,
      packetName: "VT1SUBIMAR2",
      packetObsid: 10,
      status: null,
      countedPackets: 89,
      expectedPackets: 0,
      insertionTime: null,
      updateTime: null
    },
  - {
      apid: 610,
      packetName: "MXTPHOTDATA",
      packetObsid: 10,
      status: null,
      countedPackets: 837,
```


Number of expected packets not yet available for all APIDS

This is only an example…

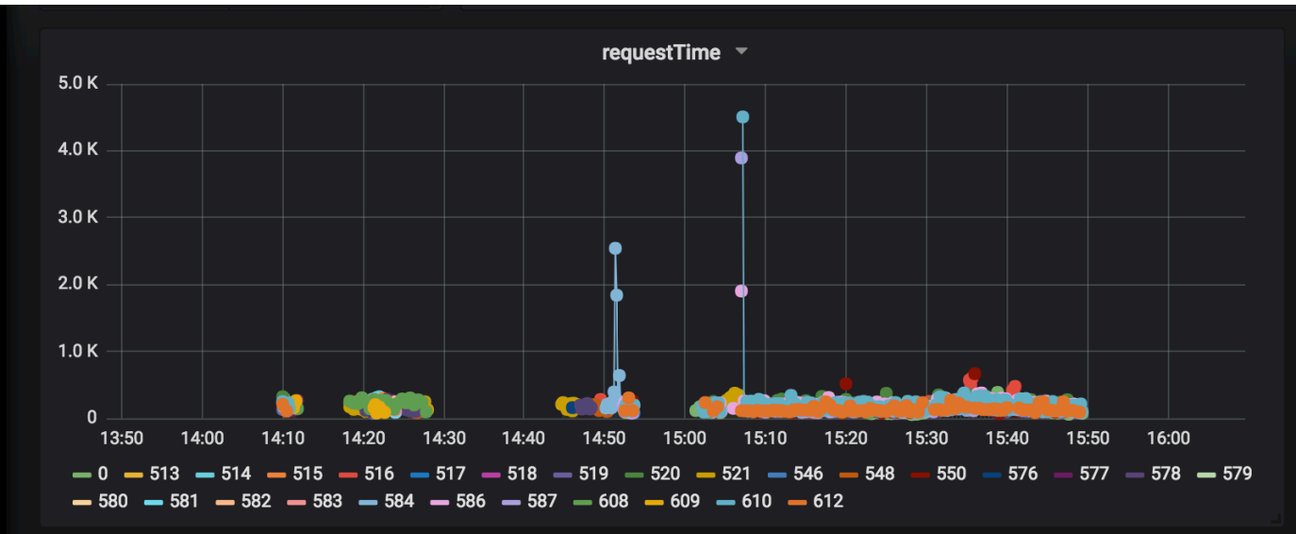
Browse data (very preliminary)

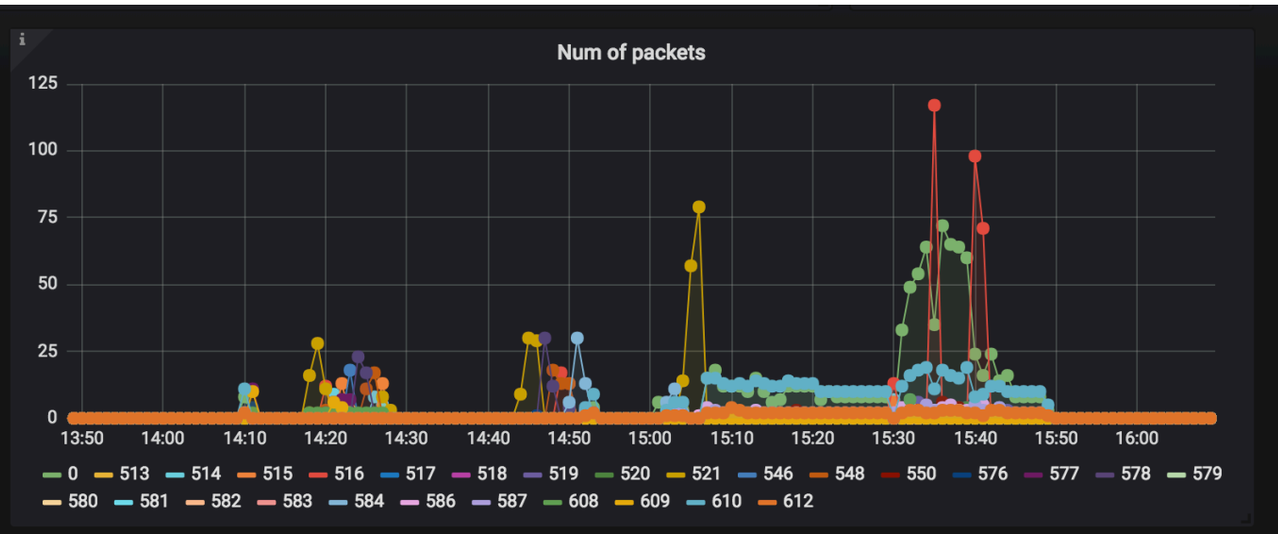
- Web UI
  - A tool to verify what the system is doing

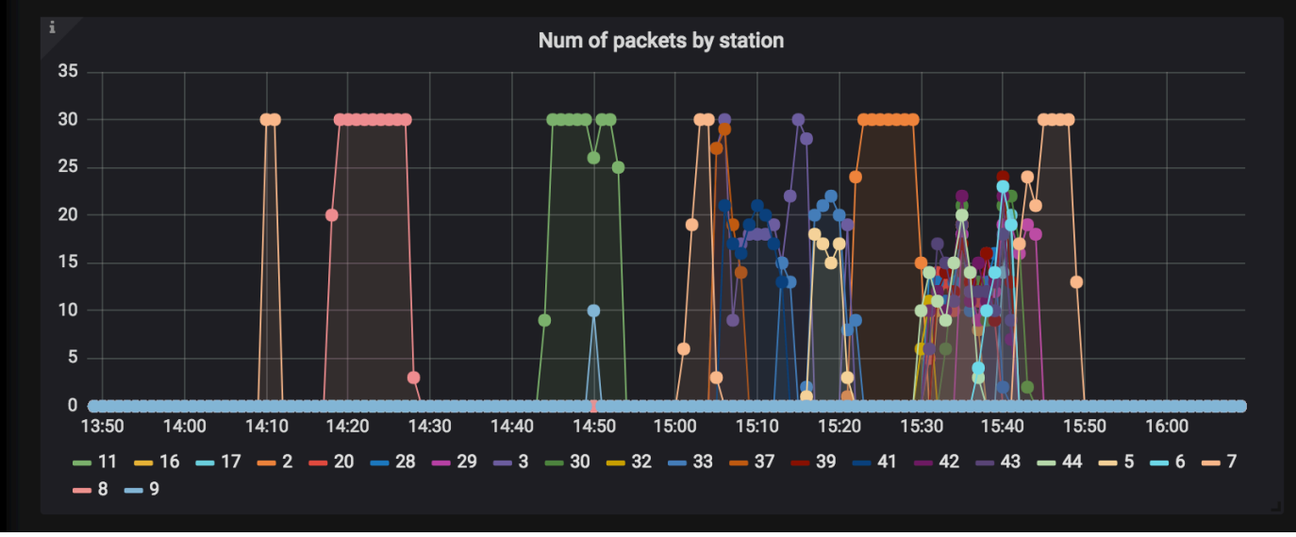


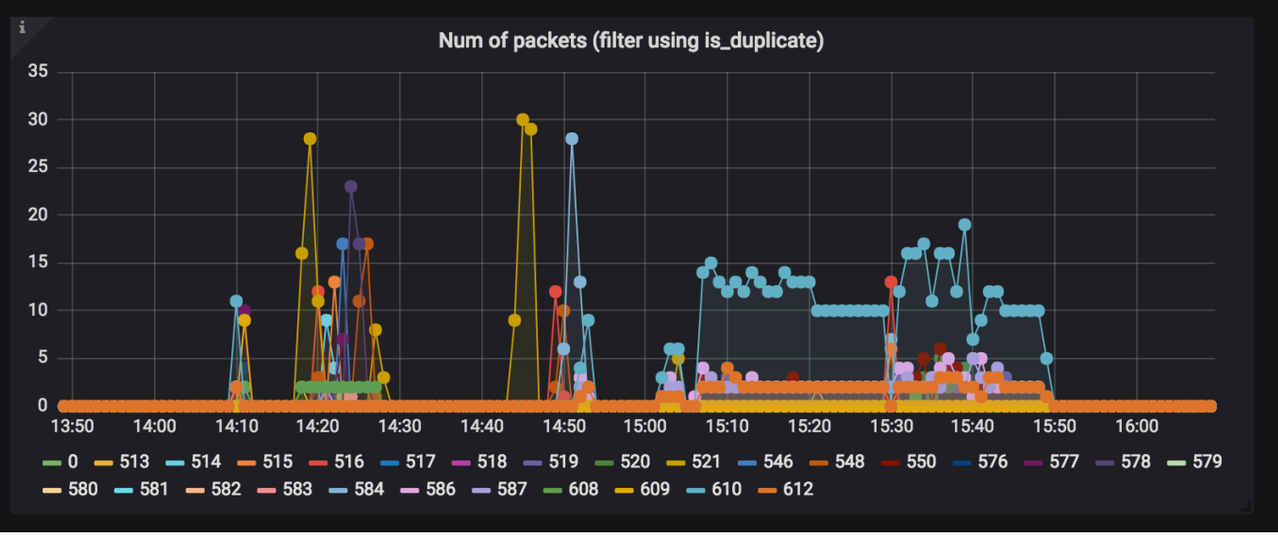
A python command line client is also available

Some plots from Grafana









Plans

- VHF data retrieval
  - Implement the final endpoints to retrieve VHF data (decoded) based on APID, OBS-ID, time…
  - Any response output here will be in JSON format
- Alert on new data being available
  - Agree on this: alerting clients means we deliver a message with the needed content to retrieve data
  - Use NATs clients for VHF (internally to FSC for the moment)