CSP pannel follow up
ObsCore extension
For radio data

F.Bonnarel

on behalf of the Radioastronomy Interest Group

CENTRE DE DONNÉES
ASTRONOMIQUES DE STRASBOURG

ESCAPE
European Science Cluster of Astronomy & Particle physics ESFRI research Infrastructures

IVOA

Follow-up of pannel discussion

- **Are you aware of the ObsCore extension for radio data ? Is that useful for discovery of the data you expose ?**
- **Access and processing of huge datasets, in which direction to go ?**
  - **Code to the data (platforms, jupyter notebooks, etc..)**
  - **SODA to extract/reshape regions of interest ?**
  - **HiPS cube for multi resolution access?**
  - **Any kind of combination ?**

Short summary of what the extension is all about

- ObsCore allows to discover datasets by constraining datasets standardised metadata
  - Instrumental Provenance (facility, instrument)
  - Identification
  - Product type (image, cube, spectrum, etc…)
  - Curation
  - Charactérisation of physical axes (spatial, spectral, time, polarisation)
  - Data access mode (url, format, datalink, cutout, etc..)
- **Is that sufficient for all kind of datasets ?**
  - **Not always**

Spatial axis addition
Uv coverage characterisation

- s_fov_min and s_fov_max (each end of the spectral window)
- s_resolution_min and s_resolution_max (each end of the spectral window)
- s_maximum_angular_scale (because large scales are filtered in interferometry)

- uv_distance_min , uv_distance_max (for scale filtering and resolution)
- uv_distribution_exc (distribution excentricity – data regularity)
- uv_distribution_fill : (distribution filling factor -data sampling)

Spectral axis additions
Product types additions

- Addition of f_resolution (as a counterpart to em_res_power)
- f_min and f_max beside em_min and em_max

- Addition of « spatial_profile » dataproduct_type
- velocity/position profiles

Sky scan modes as additional parameters tracking modes

*

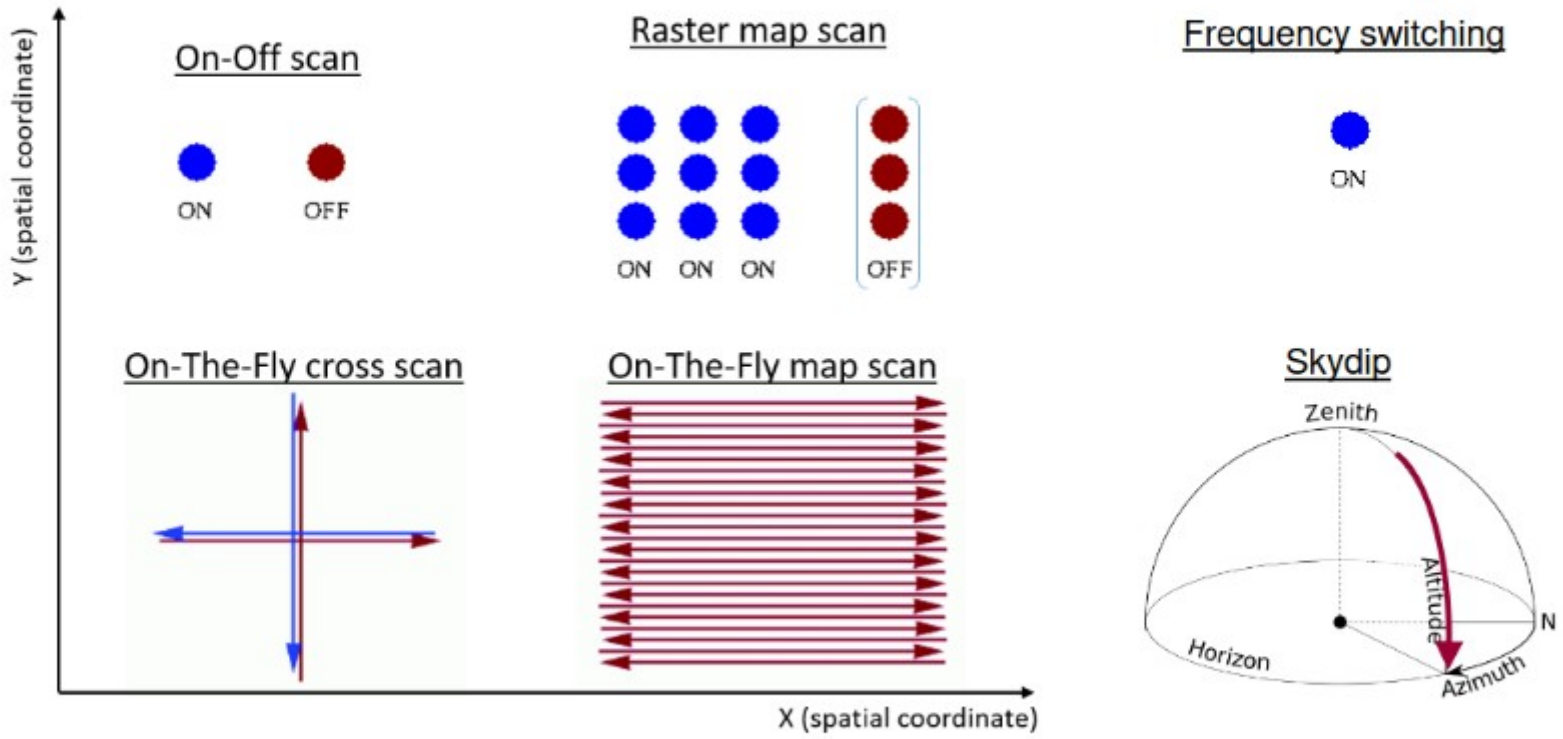


Figure 1: Single Dish Observation Sky scan modes

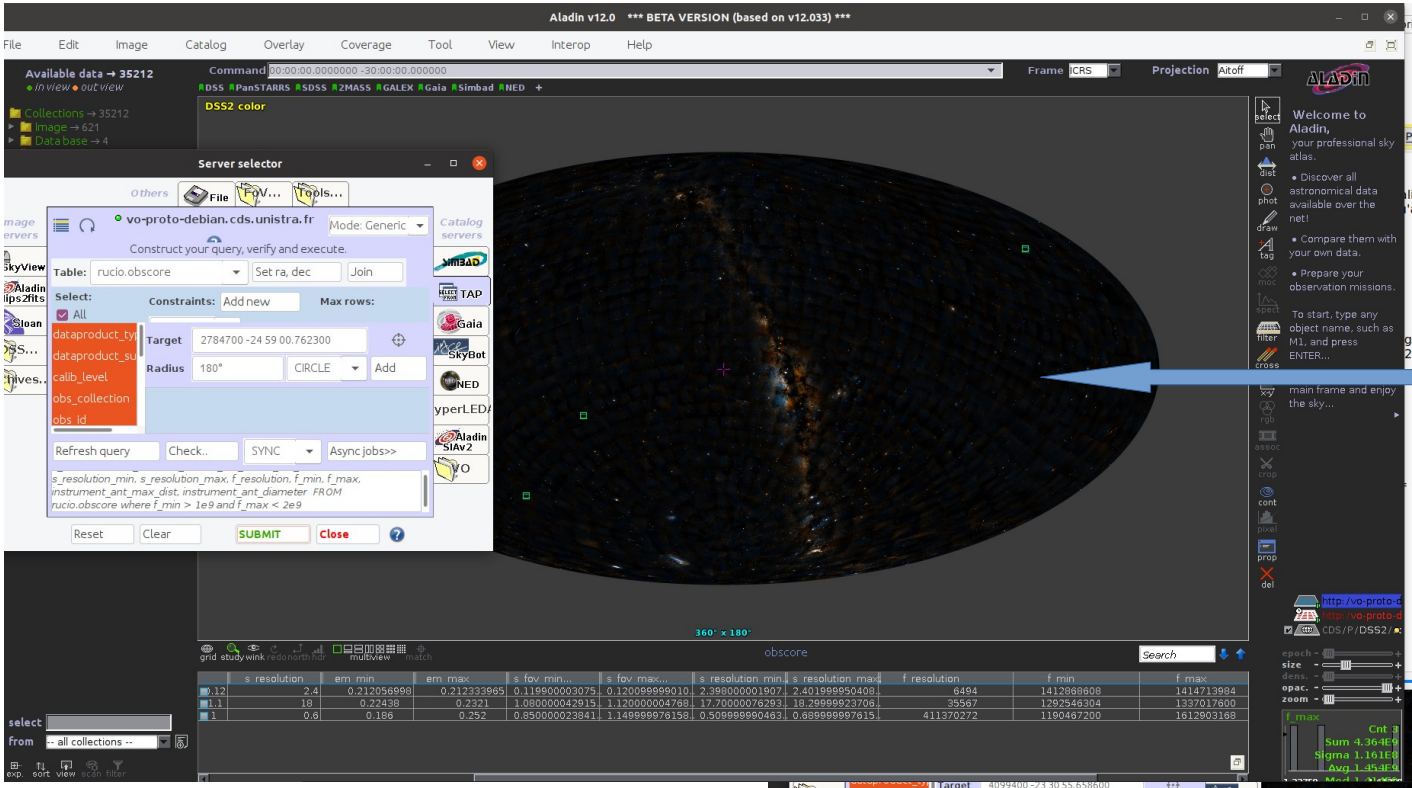
Proposed Instrumental parameters

- Antenna typical diameter (for all)
- And for interferometry
  - Number of antennae
  - Minimal distance between antennae
  - Maximal distance between antennae

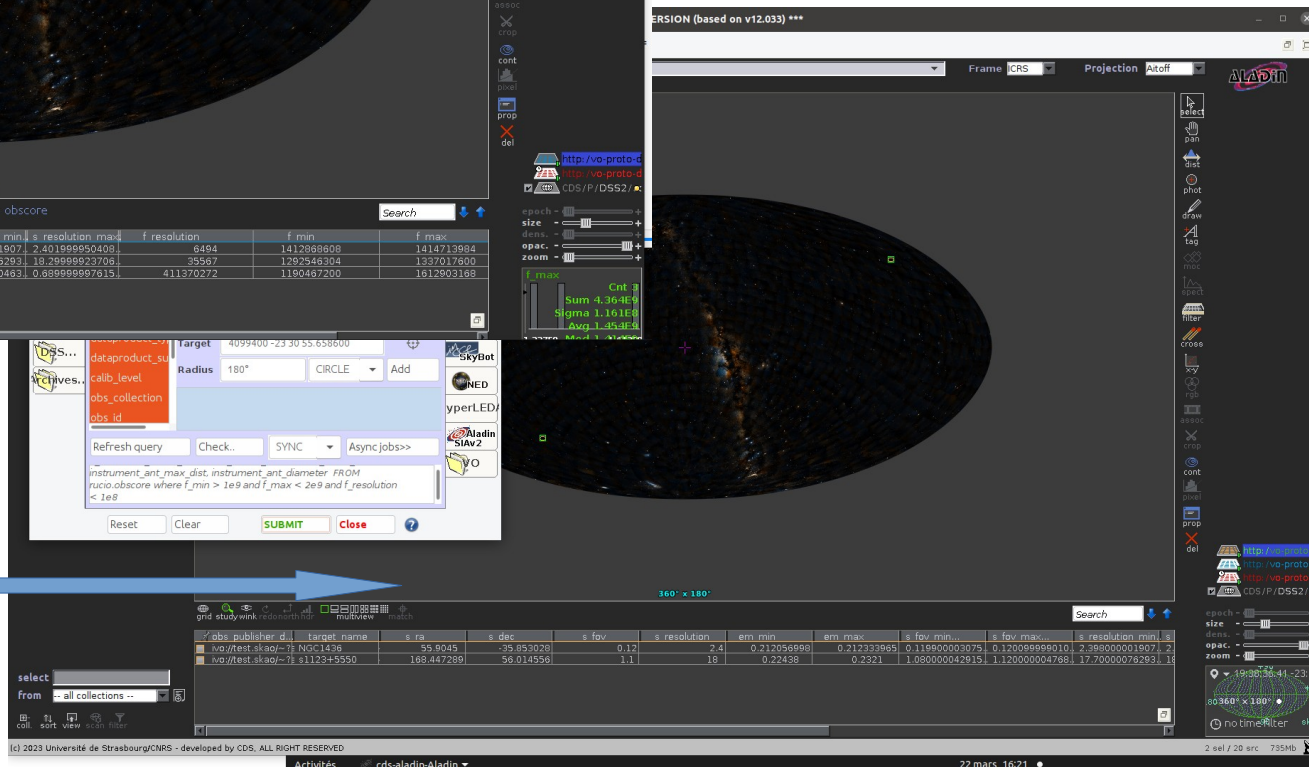
CDS prototype demo (implemented in Dachs) :

frequency between 1 and 2 Ghz (upper left – 3 results)

freq. between 1 and 2 Ghz and spectral resolution better than 100 Mhz (lower right – 2 results)



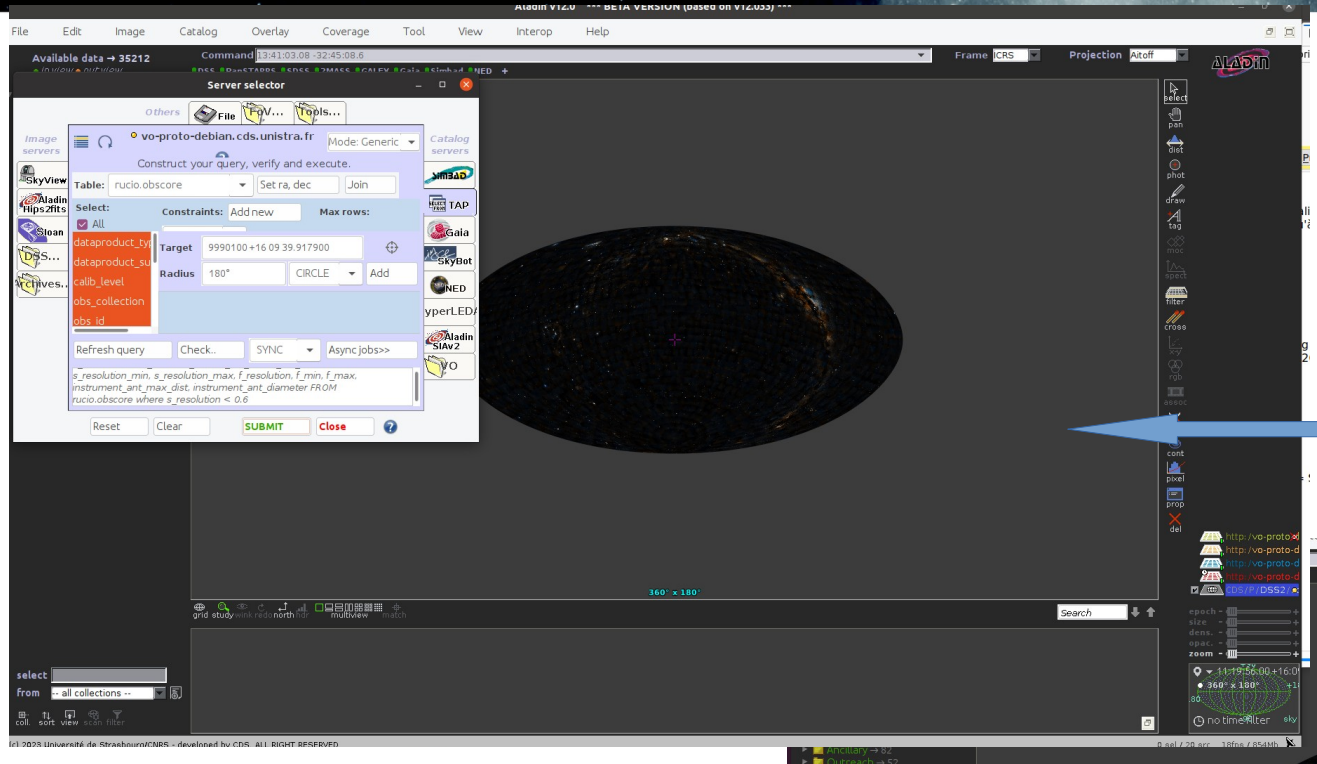
SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where f_min > 1e9 and f_max < 2e9



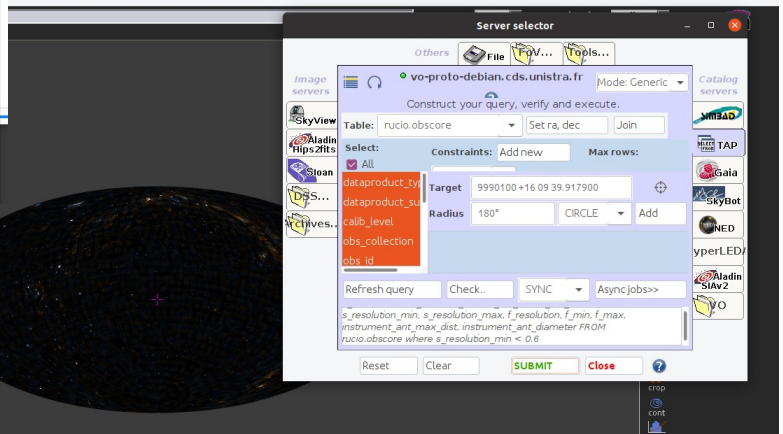
SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where f_min > 1e9 and f_max < 2e9
and f_resolution < 1e8

CDS prototype demo (implemented in Dachs) :

s_resolution better than 0.6 arcsec (upper left – no result)
s_resolution _min better than 0.6 arcsec(lower right – 1 result)



SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where s_resolution < 0.6



SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where s_resolution_min < 0.6

→ At least some part of the datsaset has a resolution better than 0.6 arcsec

CDS prototype demo (implemented in Dachs) :

s_fov larger than 7.5 deg (upper left – two results)

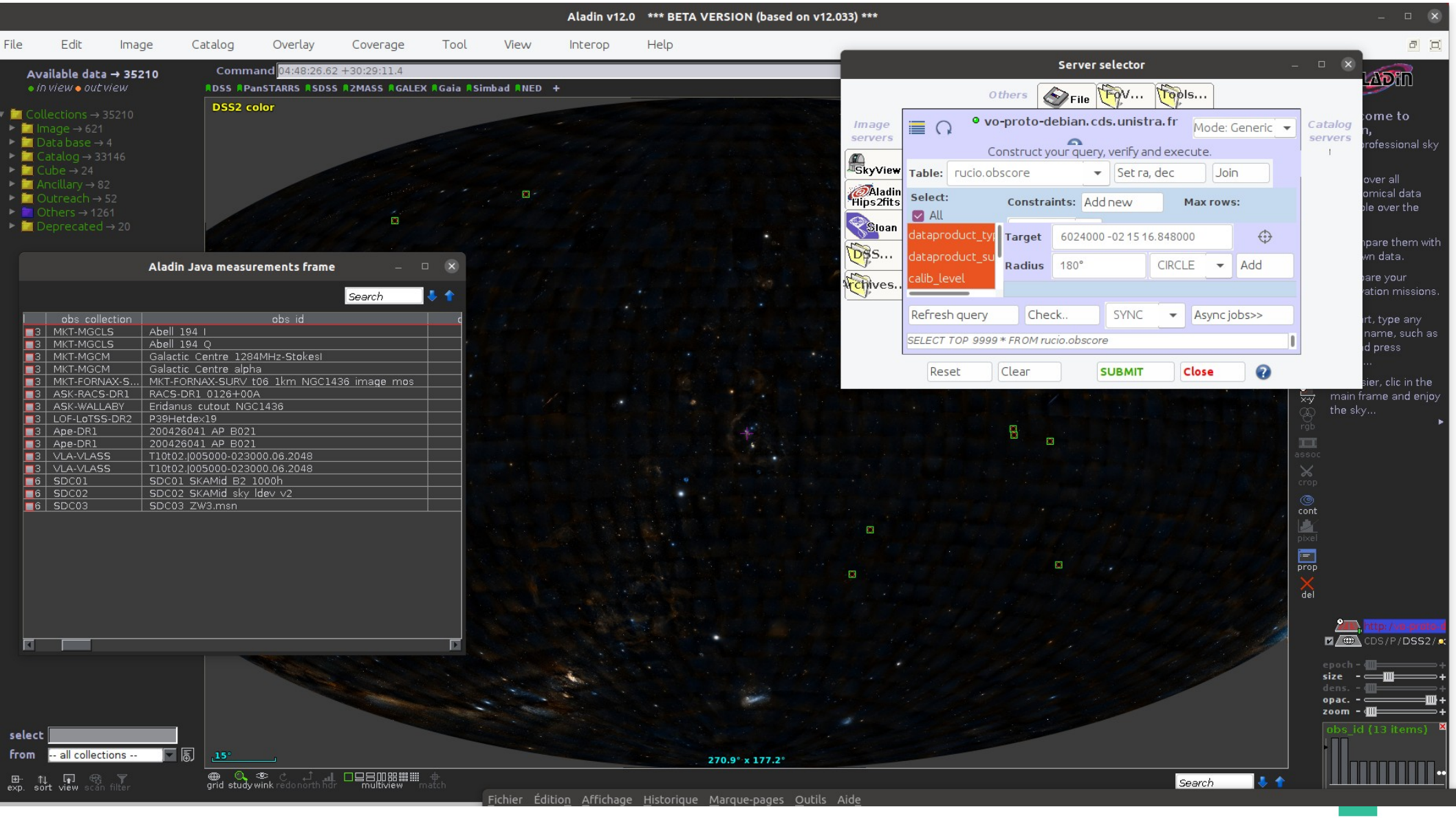
s_fov_min larger than 7.5 deg(lower right – 1 result)

SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where s_fov > 7.5 deg

SELECT obs_publisher_did, target_name, s_ra, s_dec, s_fov, s_resolution, em_min, em_max, s_fov_min, s_fov_max, s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter
FROM rucio.obscore
where s_fov_min > 7.5

→all channels of the dataset have a field of view larger than 7.5

CDS prototype demo (implemented in Dachs) :
16 datasets of ObsCore discovery service of S(ka)RCnetwork



ObsCore for complex data

- Observation made of several datasets
    - Distinction obs_id/obs_publisher_did
    - Raw per obs_publisher_did, not per observation
- Characterisation at the dataset level
- Example : raw interferometry data with
    - Main target and calibrator
    - Two different spectral windows
    - → 4 datatasets with same obs_id
- Reverse situation : dataset produced from several observations : combined obs_id

Controversy points

- f_min and f_max beside em_min and em_max
  - Rationale :
    - Something natural for the users directly available for the users
    - Something natural for the users for the queries.
    - Parameters produced from basic ones in a view
  - Cons :
    - Don’t duplicate information in the standard
    - And either :
      - Use « user defined function » in query and display
        - ivo_specconv(f, « funit », « wlunit »)
        - 1 = ivo_interval_overlaps(em_min, em_max,ivo_specconv(1.5, "GHz", "m"),ivo_specconv(1, "GHz", "m"))
      - Let the clients do the transformation in both directions

Controversy points

- Instrumental details :
    - proposed because they give an hint on some data characterization (sensibility, resolution, data quality)
    - Cons :
        - They are very specific to each experiment and do not previde generic information
        - Except (maybe) if we have use cases for that
- Science cases were missing anyway
    - Partially done (but not for instrumental details)

Controversy points

- How to expose it in a TAP service
- Two possible ways in TAP:
  - 1 basic ObsCore table, + 1 table with ObsCore+ extension. New StandardID for the latter :

    ivo://ivoa.net/std/ObsCore#table-1.0

    ivo://ivoa.net/std/ObsCore#radioExt-1.0
  - 1 basic ObsCore table + 1 table with extension only. (ivo://ivoa.net/std/obsradio#table-1.%') User or client have to join the tables
- Solution 1 become complex if we have several extensions which may be combined or not
- Solution 2 masks that the extension is meaninless wthout the baisc table. Doesn’t tell us where the baisc table lies.

Controversy points

- Compromise :
  - Set the standarID on a schema containing the Obscore table and the extension(s) with standarID ivo://ivoa.net/std/ObsCore
  - Have specific standardID on the tables (ObsCore basic, extensions)
  - Possibility to build view providing the joins in the same schema (but without standardID)