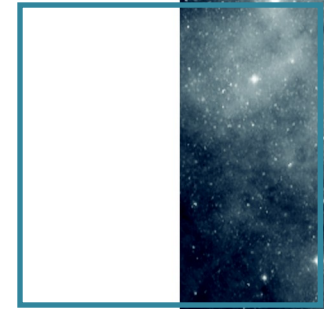


CSP pannel follow up
ObsCore extension
For radio data



F.Bonnarel
on behalf of the Radioastronomy Interest
Group



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
 - Caractérisation 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` » `dataprodukt_type`
- velocity/position profiles



Sky scan modes as additional parameters tracking modes

*

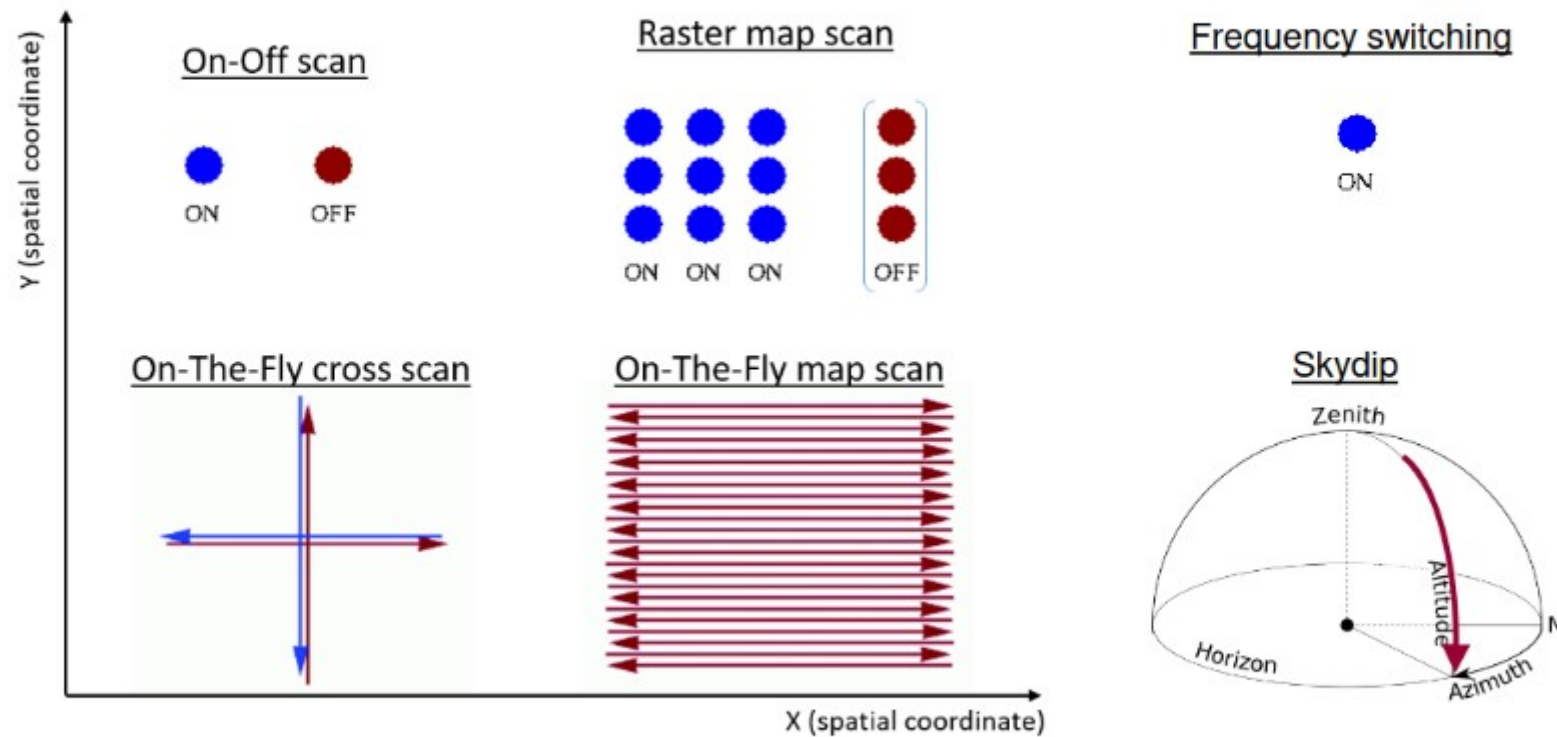


Figure 1: Single Dish Observation Sky scan modes



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)

The screenshot shows the Aladin v12.0 interface. A 'Server selector' dialog box is open, displaying a query construction interface. The query is: `SELECT obs_publisher_id, 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`. Below the dialog, a table of results is visible:

s resolution	em min	em max	s fov min...	s fov max...	s resolution min	s resolution max	f resolution	f min	f max	
0,12	2,4	0,212056998	0,212333965	0,119900003075	0,120089999010	2,39800001907	2,401999950408	6484	1412888608	1414713984
1,1	18	0,22438	0,2321	1,080000042915	1,120000004768	17,70000076281	18,29999923706	3567	192549304	1337017800
1	0,6	0,106	0,252	0,850000023841	1,149999976151	0,509999990483	0,889999976315	411370272	1190467290	1612903168

SELECT obs_publisher_id,
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_id, 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

The screenshot shows the Aladin v12.0 interface with a 'Server selector' dialog box. The query is: `SELECT obs_publisher_id, 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`.

The screenshot shows the Aladin v12.0 interface with a table of results:

obs_publisher_id	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
no/rest.skae/~? NGC1436		59.9045	-39.853028	0,12	2,4	0,212056998	0,212333965	0,119900003075	0,120089999010	2,39800001907	2,401999950408	6484	1412888608	1414713984
no/rest.skae/~? 11123+5550		169.447269	56.014556	1,1	18	0,22438	0,2321	1,080000042915	1,120000004768	17,70000076281	18,29999923706	3567	192549304	1337017800

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)

The screenshot shows the 'Server selector' window with a query: `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`. The 'Table' dropdown is set to 'rucio.obscore'. The 'Select' field is empty. The 'Constraints' field is empty. The 'Max rows' field is set to 'All'. The 'Refresh query' button is highlighted. The main window shows a dark sky image with a red crosshair. A blue arrow points from the text above to the 'Submit' button.

```
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 dataset has a resolution better than 0.6 arcsec

The screenshot shows the 'Server selector' window with a query: `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`. The 'Table' dropdown is set to 'rucio.obscore'. The 'Select' field is empty. The 'Constraints' field is empty. The 'Max rows' field is set to 'All'. The 'Refresh query' button is highlighted. The main window shows a dark sky image with a red crosshair. A blue arrow points from the text above to the 'Submit' button. Below the main window, a table of results is visible:

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
0.188	0.252	0.352000923841	1.14969997915	0.3069999904632568	0.6899999976158142	4.11370272	1169467200	1017393188	75266	12.53699999

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.obscure  
where s_fov > 7.5 deg
```

```
FROM rucio.obscure  
where s_fov > 7.5
```

Aladin v12.0 *** BETA VERSION (based on v12.033) ***

Command: DSS2 color

Server selector dialog:

- Table: rucio.obscure
- Constraints: Add new
- Max rows: []
- Select: All
- Target: 1872200 -19 18 33.425300
- Radius: 180°
- Shape: CIRCLE

Query: `s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter FROM rucio.obscure where s_fov > 7.5`

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
ivo/test.skao/~71	0126+00	21.72206541667	0.00342372222	9.5	25	0.338	0.338	9.5	9.5	25	25
ivo/test.skao/~71	0.0-30.0	0	-30	9.1	16	1.53	2.83	6.3800011444	11.8100004196	11.22989954223	20

```
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.obscure  
where s_fov_min > 7.5
```

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

Aladin v12.0 *** BETA VERSION (based on v12.033) ***

Server selector dialog:

- Table: rucio.obscure
- Constraints: Add new
- Max rows: []
- Select: All
- Target: 2957000 +00 00 12.325400
- Radius: 9.899°
- Shape: CIRCLE

Query: `s_resolution_min, s_resolution_max, f_resolution, f_min, f_max, instrument_ant_max_dist, instrument_ant_diameter FROM rucio.obscure where s_fov_min > 7.5`

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
ivo/test.skao/~71	0126+00	21.72206541667	0.00342372222	9.5	25	0.338	0.338	9.5	9.5	25	25

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

The screenshot displays the Aladin v12.0 software interface, which is a BETA VERSION based on v12.033. The main window shows a star field with a command line at the top: `04:48:26.62 +30:29:11.4`. The interface includes a menu bar (File, Edit, Image, Catalog, Overlay, Coverage, Tool, View, Interop, Help) and a toolbar. On the left, there is a sidebar with 'Available data → 35210' and a tree view of collections. The main display area shows a star field with a 'DSS2 color' overlay. In the bottom left, there is a table titled 'Aladin Java measurements frame' with columns 'obs collection' and 'obs id'. In the bottom right, there is a 'Server selector' dialog box with a query builder interface. The query builder shows a table 'rucio.obscore' and a query: `SELECT TOP 9999 * FROM rucio.obscore`. The query builder also includes fields for 'Target' (6024000 -02 15 16.848000) and 'Radius' (180°).

Aladin v12.0 *** BETA VERSION (based on v12.033) ***

File Edit Image Catalog Overlay Coverage Tool View Interop Help

Available data → 35210
in view out view

Command 04:48:26.62 +30:29:11.4
DSS PanSTARRS SDSS 2MASS GALEX Gaia Simbad NED +

DSS2 color

Aladin Java measurements frame

obs collection	obs id
MKT-MGCLS	Abell 194 I
MKT-MGCLS	Abell 194 Q
MKT-MGCM	Galactic Centre 1284MHz-StokesI
MKT-MGCM	Galactic Centre alpha
MKT-FORNAX-S...	MKT-FORNAX-SURV t06 1km NGC1436 image mos
ASK-RACS-DR1	RACS-DR1 0126+00A
ASK-WALLABY	Eridanus cutout NGC1436
LOF-LoTSS-DR2	P39Hetdex19
Ape-DR1	200426041 AP B021
Ape-DR1	200426041 AP B021
VLA-VLASS	T10t02.i005000-023000.06.2048
VLA-VLASS	T10t02.i005000-023000.06.2048
SDC01	SDC01 SKAMid B2 1000h
SDC02	SDC02 SKAMid sky ldev v2
SDC03	SDC03 ZW3.msn

Server selector

Others File FOV... Tools...

Image servers

vo-proto-debian.cds.unistra.fr Mode: Generic

Construct your query, verify and execute.

Table: rucio.obscore Set ra, dec Join

Select: Constraints: Add new Max rows:

All

dataprod... Target 6024000 -02 15 16.848000

dataprod... Radius 180° CIRCLE Add

calib_level

Refresh query Check.. SYNC Async jobs>>

SELECT TOP 9999 * FROM rucio.obscore

Reset Clear SUBMIT Close ?

select
from all collections --

15°
270.9° x 177.2°

Fichier Édition Affichage Historique Marque-pages Outils Aide

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 datasets 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 (sensitivity, resolution, data quality)
 - Cons :
 - They are very specific to each experiment and do not provide 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 meaningless without the basic table. Doesn't tell us where the basic table lies.



Controversy points

- Compromise :
 - Set the standardID on a schema containing the Obscore table and the extension(s) with standardID `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)

