





# How to Discover and Query Obscore Extensions

Markus Demleitner HE in the VO workshop Paris, 2025-04-30

#### **Discovery of Extension Tables**

This ought to be as per TableReg http://ivoa.net/documents/Notes/TableReg/: Give me URLs of (auxiliary) TAP services for

- vs:CatalogResource-typed records
- with a table with utype matching a bespoke pattern

### Why?

- A separate record (as opposed to the TAP service's record) so we have somewhere to keep our coverage metadata in
- ...and also isServedBy-relationships from data collections we keep in the table.
- CatalogResource so we don't match the embedding TAP record (which is a CatalogService).
- utype because we are identifying something DM-like; pattern because sometimes we want to see minor versions but in general we don't care about them.

## In ADQL (for obs\_radio)

```
SELECT DISTINCT table_name, access_url

FROM rr.res_table

NATURAL JOIN rr.capability

NATURAL JOIN rr.interface

NATURAL JOIN rr.resource

WHERE

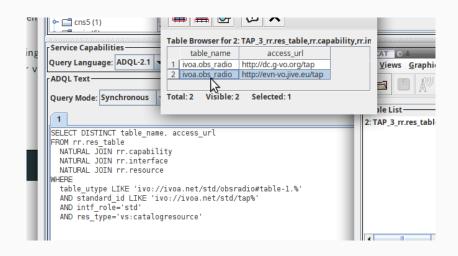
table_utype LIKE 'ivo://ivoa.net/std/obsradio#table-1.%'

AND standard_id LIKE 'ivo://ivoa.net/std/tap%'

AND intf_role='std'

AND res_type='vs:catalogresource'
```

#### It works!



#### The Identifier

Standards should in general only deal with their own registry records. Hence, each extension's utype should have the form

ivo://ivoa.net/std/obs<whatever>-table-<major>.<minor>

Let's not repeat the mistake of ADQL 2.1 that defines lots of keys in TAPRegExt's StandardsRegExt's record.

#### Global Disovery in pyVO

With an experimental pyVO registry constraint, running an obs\_radio query globally would look like this:

```
results = []
for res in pyvo.registry.search(HasRadio()):
    svc = res.get_service("tap")
    print(svc.baseurl)
    results.append(svc.run_async(
        "SELECT TOP 1000 t_min, s_resolution_min, access_url"
           FROM ivoa.obs_radio NATURAL JOIN ivoa.obscore").to_table())
union = vstack(results)
```

See the attached full source for more details:

