A New Registry Interface Proposal VO-Paris

Jonathan Normand, Pierre Le Sidaner
Observatoire de Paris

Registry: current interface

- All resources are defined by a xml schema

■ Search method on defined fields

■ Keyword search on (identifier, content/ description,title, @xsi:type, content/subject)

Both methods use complex ADQL-1 language over SOAP

Difficulty to query, not all registry respond, very slow

In fact some full searchable registries use Xpath, others use an internal protocol

The Evolution proposed

Only define the service behavior, not the implementation.
Looks like open search

- REST access using SEARCH method

`http://<my_url>/search?q=text[&text]`

Plain text search in the list of fields

Example

`http://voparis-registry.obspm.fr/registry/_search?q=infrared`

REST access using serach by field

http://<my_url>/search?q=field:text1[&field2:text2]

search on specific field

Example

```
voparis-registry.obspm.fr/registry/_search?q=standardid:
"ivo://ivoa.net/std/ConeSearch”&q=description:"infrared"&
default_operator=AND
```

The Evolution proposed

Focus of search operation: (All theses fields can be put in the query)

- subjects
- title
- shortname
- description
- type
- referenceurl
- publisher
- contactname
- capabilities[accessurl, standardid]
- identifier
- registryid
- updated
- created
- status

NB: Even if capability is a table, a search can be done on that field

```
http://voparis-registry.obspm.fr/registry/_search?q=
standardid:"conesearch"
```

The Evolution proposed

Returns all the parameters + access URLs of the VOResource

The return format is only JSCON for now

If the interface is accepted, a XML list of embedded VOResources could be added

The implementation can directly handle geographical queries using a geojson description of the resource

The registry should contain MOC information and a separate service can handle this capability

Implementation for validation

- All classical services CS, SSA, SIA have been ingested in a no-sql database couchdb, with the field of research (capabilities, description, identifier, subject, type).

+ easy to modify because its structure is not fixed
+ easy to maintain
+ easy to ingest new resources (indexed on the fly)

For implementing the search method, the search engine ElasticSearch (build on top of Apache Lucene) has been used.

+ Really powerful, quick and adapted to text search.
+ Can face increase of resources.
+ Scalable

Example to play

I want to have all TAP services

http://voparis-registry.obspm.fr/registry/_search?
q=standardid:”ivo://ivoa.net/std/TAP”

or simply

http://voparis-registry.obspm.fr/registry/_search?
q=”ivo://ivoa.net/std/TAP”