TAP Service Discovery in TOPCAT

Mark Taylor (Bristol)

IVOA Interop
Shanghai
16 May 2017
TAP service discovery

- The problem
- Current solution: GloTS
- Possible solution: Aux registry records
- Pros and cons

See also talk from *Sydney 2015* for more background
TOPCAT TAP service discovery requirements

- Users want to use TAP
  1. Identify which TAP service to use
  2. Make queries etc using that service

- Desirable user searches:
  ▶ By Service metadata
    - Search by Service Name, IVOID, Publisher, Description, ...
    - List matching services
    - Works well with the Registry
    - Less common use case

  ▶ By Table metadata
    - Search by Table Name, Description, Authors, Spatial/Spectral coverage, ...
    - List matching tables within services
    - Problems with the Registry
    - More common use case
Table Metadata

Where is the table metadata?

- In the TAP service registry record?
  - Tables can be declared as part of the TAP service registry record
  - Searching in that case is easy
  - Some services do store it like that ...
  - ... but others do not
  - ... and at least in some cases that’s a good thing (TAPVizieR > 30 k tables)

- Elsewhere in the registry?
  - Individual tables can be registered as their own resources
  - ... in some cases this is done already (e.g. all VizieR’s Cone Search services)
  - ... in others, service providers would have to do it specially

- Available from the service itself
  - TAP services provide metadata query facilities (TAP_SCHEMA and usually /tables)
  - It’s not feasible to query all TAP services for service discovery
  - So this requires some central component to query all services for metadata, and maintain this aggregated metadata in a searchable form
  - GloTS does this! Thanks Markus.
Table search using GloTS

Global TAP Schema

- Non-standard TAP service run by Markus at ARI (GAVO DC TAP)
- Auto-updated by regularly crawling TAP services and reading metadata (?)
- It contains info on all registered TAP services in 3 tables:
  - `glots.services` (123 rows)
  - `glots.tables` (39k rows)
  - `glots.columns` (227k rows)

- Pros:
  - Easy to interrogate
  - Fast
  - Service is in practice reliable

- Cons:
  - Doesn't support all imaginable queries (e.g. by Author/Creator, coverage, ...)
  - Non-standard
  - Single point of failure (+ mirrors)
  - Requires maintenance effort (by Markus)
  -Duplicates data
  - Bypasses Registry
Note: Discovering Data Collections within Services (Demleitner+)

- Proposal (as concerns TAP 1.0):
  - Mark table-records-outside-TAP-service-records with
    
    \[
    \text{<capability standardID="ivo://ivoa.net/std/tap#aux">}
    \]

- Enables:
  - Service enumeration queries pick up all TAP service records:
    
    \[
    \text{WHERE standard_id = 'ivo://ivoa.net/std/tap'}
    \]
  - Data discovery queries pick up all tables inside/outside TAP service records:
    
    \[
    \text{WHERE standard_id IN ('ivo://ivoa.net/std/tap',}
    \]
    \[
    \text{'ivo://ivoa.net/std/tap#aux')}\]

- Actually it’s a bit more complicated than that
  - ... to cope with Identifiers v2 and minor version pattern matching

For TOPCAT:

- Only Data Discovery queries are useful
  (Service Enumeration gives you services with no information about their tables)
- Still need to find which service each discovered table belongs to
Requirements:

- Pick up table metadata within TAP service records ivo://ivoa.net/std/tap
  - Tables with capability `standard_id='ivo://ivoa.net/std/tap'`
  - TAP service is parent resource
- Pick up table metadata outside TAP service records (Cone Search or standalone)
  - Tables with capability `standard_id='ivo://ivoa.net/std/tap#aux'`
  - Need to identify TAP service
- Select by metadata constraints on both lists, join for presentation to user
Table search using Registry: Theory

Requirements:

- Pick up table metadata within TAP service records ivo://ivoa.net/std/tap
  - Tables with capability standard_id='ivo://ivoa.net/std/tap'
  - TAP service is parent resource

- Pick up table metadata outside TAP service records (Cone Search or standalone)
  - Tables with capability standard_id='ivo://ivoa.net/std/tap#aux'
  - Need to identify TAP service

- Select by metadata constraints on both lists, join for presentation to user

How to identify TAP service for external tables?

- Use rr.relationship table (relationship_type='served-by')?
  - Looks like the right thing to do ...
  - ... but relationships often not recorded

- Match capability access_url against known TAP service URLs?
  - Seems somewhat fragile (access_url not intended as key)
  - Access URLs sometimes differ in practice (probably fixable)
Table search using Registry: Implementation

Implemented in TOPCAT (v4.4+ pre-release & next public release)

- Choose between **GloTS** (default) and **Reg Prototype** discovery options
- It works ... for some services

```
SELECT
  ivoid, table_name, table_description
FROM (
  SELECT DISTINCT
    table_name, table_description, access_url
  FROM rr.res_table
  NATURAL JOIN rr.capability
  NATURAL JOIN rr.interface
  WHERE standard_id LIKE 'ivo://ivoa.net/std/tap%'
  AND 1=ivo_nocasematch(table_name, '%tgas%')
) AS tbl
JOIN (
  SELECT DISTINCT
    ivoid, access_url
  FROM rr.resource
  NATURAL JOIN rr.capability
  NATURAL JOIN rr.interface
  WHERE standard_id = 'ivo://ivoa.net/std/tap'
) AS serv
USING (access_url)
```

Run `topcat -verbose -verbose` to see the ADQL on stderr.

Mark Taylor, Service discovery in TOPCAT, Interop, Shanghai, 16 May 2017
Conclusion

GloTS vs. Registry+Aux from TOPCAT/TAP point of view

- GloTS:
  - Works great!
  - Maintenance burden (on Markus)
  - Non-standard, sidesteps/duplicates IVOA/OAI-PMH practice
  - Centralised (single point of failure)
  - Precludes certain searches

- Registry with #aux records:
  - Proof of concept works
  - Requires (modest?) effort from data providers; currently only a few comply
    - Unregistered TAP tables must be registered (in or out of TAP service record)
    - Tables registered for Cone Search must be marked additionally tap#aux
  - Standalone table → TAP service link is weak
    - Require also proper served-by→TAP relationship records?
    - Or some other option (on-behalf-of attribute of capability element?)

- Conclusion
  - I don’t mind which, as long as it works
  - Registry looks like a better/more respectable long term solution ...
  - ... if data providers can publish appropriate records
    - Needs wider testing by data providers before adoption (please prototype!)
    - Inclusion in TOPCAT TAP list might be a big enough lever?