Implementing a TAP service at the HEASARC: Lessons Learned

Tom McGlynn

(Preliminary)

What was done

- ADQL v2.0 (- region, uploaded tables)
- PQL v0.2 (-multicone)
- UWS (-abort)
- VOSI (-capabilities)
- No version negotiation
- Still VOTable 1.1
- No VOSpace support

Standards used within TAP implementation

- TAP 1.0
- VOTable 1.2
- UCD 1.23
- UWS 1.0
- VOResource 1.1
- VODataService 1.2
- VOSI 1.0
- ADQL 2.0
- PQL 0.2 (optional)
- STC 1.33
- STC-S (only defined in notes, not standards)
- *VOSpace 1.15 (optional)*

Many standards are strongly coupled
Intimidating even for an experienced developer of IVOA protocols.

Core TAP functionality

- Standard way to query astronomy databases.
- ADQL, PQL interfaces (no geometry types)
- Standard TAP keywords
- UWS (or direct HTTP) for query
- VOTable for results

Issues, but generally implementable

Discovery Support

- VOSI, VOResource, VODataService
- Version negotiation
- TAP_Schema

Some unresolvable issues and duplication of functionality

Distributed access

- Remote table access
- Table upload
- VOStore

Untested, some issues

Advanced Queries

- ADQL geometry types
- Regions in ADQL and PQL

Minimal requirements ill-defined so hard to understand what must be implemented.

Specific Issues

VOSI: Capabilities

- TAP resources are supposed to describe their capabilities both directly and within the registry.
  - TAP implementations have many choices for what to implements (e.g., PQL, REGION level, coordinate systems,…) so this is actually important!
- No definition for how to do this
  - And it's very hard to track this down. 2.2.3 suggests ref 8 (VOResource) which in turn references VODataService, but none of these seem to say anything useful. Very hard to read without guidance.
    - Extremely frustrating.
  - TAP needs to define interface with VOSI cleanly.
- 2.2.4 says capabilities optional unless async is implemented when it becomes mandatory. But async is mandatory so…
- Curious restriction to HTTP GET only? Why phrased negatively "This resource supports only the http GET method" as opposed to positively. "This resource MUST be accessible through the HTTP GET method." [Other VOSI elements too.] Overcoupling with HTTP? Am I required to fail if when POST is used?
- An example document would go a long way to resolving issues.

- Decided to forego any support for capabilities.

ADQL REGIONs and Coordinate Systems

- No (easily found at least) statement of what REGION strings are supposed to be.
  - STC-S not defined in ref [4] (STC) which makes reference to a non-standard note to define STC-S. Current version of this document was not in VO repository.
  - Even there what's allowed in region is not clear.
  - ADQL document gives no clue (in 2.4.14) either.
- What are the levels of geometry support that are allowed in a conformant TAP implementation?
- Can I support Circle/Box/Polygon/Point without REGION?
  - What are region 'constructs' as opposed to REGIONs. [ADQL treats CIRCLEs and REGIONs as different types. It uses the terminology 'geometry types' or 'geometries' never (I think) 'region types' or 'region constructs'.
  - Another implementer suggested no, region types includes CIRCLES, boxes, ... But this is never stated.
  - REGIONs are much harder than other geometry types: natural break point to support others only.
  - What does it mean to say '… the extent of STC-S support within the region function…' is left up to the implementation. Is 'No STC-S is supported' OK so that REGION support is actually optional?
- How is the extent of region support reported to the user?
- 2.3.4 "Coordinate system specification (sic) for … must use values from Table 3 STC."
  - Typo? made me wonder if this was intended to suggest that other values were fine but I needed to support at least some values from Table 3.
- How is the extent of coordinate system support reported to the user?
- Examples would help a lot.

- Decided to forego supporting REGION and support only ICRS and Galactic coordinates.

VOTable-ADQL table

- General
  - TAP goes from Database-User. ADQL and PQL are ways to do queries, but do not define underlying types.
    - Placement in document. 2.5 is an optional capability. Should be in mandatory element of document, maybe right up front.
    - Needs to be identified since it is critical and should probably be referenced by language documents.
  - Not clear what ADQL types are – ADQL doesn't support CREATE TABLE. Need to worry about SQL types or more generally types in the underlying database.
    - ADQL document mentions types: double and integer in functions, and the geometric types Point, Circle, Box, Polygon and Region (and distinguishes all of these types)
  - Does not address more fundamental issues of how to translate internal data formats to VOTable (or other) outputs.
    - How is a boolean (SQL99) column to be represented?
      - Our real databases are not limited to SQL92.
    - User defined types are increasingly common.
    - …
    - Given that exhaustive support for all types may not be possible, recommendation for how to represent unspecified values.
- Uploads
  - What happens when VOTable is ingested and type or column name is not compatible with database? [Array type, name has spaces, …] Lots of options including error, but don't know which to do. May be very common.
  - No discussion of structure of VOTable – can I access second table of complex VOTable?
- Limited impact on HEASARC because we use very simple types, but I think this reflects a misconception of relationship of TAP and ADQL. 'ADQL talks to TAP tables', not 'TAP takes ADQL results'. TAP is the master!
- Table uploads not implemented.

Timestamp literals

- Peculiar interjection of support for particular format for ‘timestamp’ literals (2.3.4, 2.3.5)
  - ‘the service must support the use of timestamp values in the ISO format’
    - Definition of timestamp literal value: in what contexts is this in effect, e.g.,
      Does **Select ‘1990-10-10’** return a string or a timestamp? How about
      Select ‘1990-10-10’ union select x from timestamp_column
  - Why in TAP document and not in ADQL/PQL?
  - ADQL/PQL don’t know anything about TimeStamp columns per se either.
  - Distinction between support for this format in VOTable representation versus in query.
- What types are really meant?
  - SQLServer timestamps are not dates at all.
  - Do we mean to include dates that are stored in alternative formats?

- Lots of systems seem to drop T separator. Is it mandatory?

- HEASARC has all dates in MJD ints or doubles. PQL interface allows ISO format times or ranges, ADQL does not, and it would require a significant effort to add. Requiring a function datetime(string) would be much easier and much less ambiguous (and maybe meets requirement) but really seems like it belongs in language standards.

ADQL namespace

- Several references to ADQL name space (xtype for timestamp and region values)
- Must be defined to create valid XML documents, but no definition found.
- Needs to be consistent between VOTables so can’t just punt.

- But so far I have…

Version negotiation

- Do we need this in first version?
- What is the version? Where does a user find out what version is? Not specified.
  - Actually say what the version string is!
  - Does this whole issue belong in VOSI?
- How do we handle upward compatibility?
  - Could a 1.1 client ask for 1.1 data from a 1.0 Service?
  - Could a 1.0 client ask for 1.0 data from a 1.1 service without version support?
  - If neither then what does this mean about IVOA compatibility rules.
- Relationship of TAP version with version of related protocols.
- What's 'level two'? 2.8.3 'If the client specifies … and this does not match a version available from the service *at level two,* the service …'

- Requirement ignored in HEASARC implementation.

UWS incompatibility

- UWS standard says that results are named and user should get URI associated with a given name.
- TAP standard says user should get result at a specified URL.
  - Only in what is labeled as non-normative section of the standard (5.2)
- This part of UWS is also being looked at.
- Use of Error attribute to point to VOTable also pushes bounds of UWS standard where we have:
  - 'The error object gives a human-readable error message for the underlying job'
  - XML is not really human readable.
  - Somewhat inconsistent with Sync where the result includes the error. If PHASE=error does results/result also point to the error document?

- Not a barrier to implementation, but an inconsistency between standards. Need to define results in normative section of document.

TAP metadata

- Three ways to get metadata
  - VOSI, TAP_Schema, MAXREC=0
- What is the relation of these?
- What metadata is unique to each? How do they map to each other?
- If we have MAXREC=0, do we need to run the query in case it might not return any records (so as to give the correct overflow indicator). If so, is MAXREC=0 special (the document says it is). What about MAXREC=-1?

- This was rather confusing to implement and finding the documentation was harder than needed. Issues are more for clients than servers. Examples!

What are the geometry classes?

- Table in section 4 suggests that we have Point and Region but not Circle, Box and Polygon.
  - Where is this specified?
  - Some ADQL functionality (e.g., centroid) might only work on some.
    - (e.g., what is centroid of a polygon?)
  - Why is Point distinguished?
- UCDs for geometry columns.
- TAP doing work that may belong to ADQL?

- So far HEASARC has punted.

TAP Schema

- Redundancy in Schema for Tables table.
  - Has both schema name column and table_name column which says that it is fully qualified name.
    - Do I put in schema twice? What if this is ‘default’ schema?
  - Can default schema be left blank?
- Joins
  - Do join tables need to reflect themselves, i.e., are they allowed to be empty.
  - Many-many joins

- Didn’t repeat schema in Tables. Didn’t use join tables (i.e., they are empty). Examples!

Overview in 1.4 is too condensed

- The non-normative section 1.4 is too dense. Expand and clarify.

- This is the only section that I felt gave me any feel for TAP as a whole but I felt overwhelmed by the amount of material covered. A much expanded and more comprehensive section would be invaluable, especially – I think – for builders of TAP clients.

REQUEST parameter in Async Requests

- Current language strongly suggests that REQUEST parameter needed for all UWS requests.
- Needs to be clarified.

- Not a real problem for implementation.

Columns returned

- 2.7.1 seems too ADQL specific in terms of rows returned. [No SELECT is required in PQL queries] What about select *?
- What is this really trying to say? Should not add or reorder columns?

- Ignored (but I think implementation satisfies it as best I understand it).

Musts in section 7

- Described as informative, but includes many musts. We don’t get to define HTTP.