IVOA Nov 2024 DAL

Time: Sat 16 Nov 09:00 UTC+1

DAL at IRSA *(Anastasia Laity)*

- draft ivoa.obscore based off CADC's implementation

Simulated Data in VO Services

A need to distinguish simulated versus real data

- It is possible for a user to accidentally request simulated data
- solution; only serve simulated data through dedicated endpoints
- ivoa.obscore mapped to hidden simulated_obscore table

Users might forget they fetched simulated data

- An extra column, irsa_provenance=SIMULATED ALWAYS included
- Additionally, a "simulated" prefix is always added to column data, whenever possible

ObsTAP / Datalink Status

- Using CAOM on the backend, each row will produce a group of related artifacts/files
- Each row is a CAOM.plane row
- For datalink: each row = caom.artifact

Barriers to Release

- #1 is performance:
    - Can't easily fix with better indexing / data types
    - Supply planeid directly (hashed thanks to an additional parameter named FASTID=hash) as service input for improvement - after an SIA/ObsTAP Query
    - However, possibly not repeatable -- id's are not the same -- need an expiration
- SIA2 rollout plan
    - Changing to a link to datalink table; need to help clients with transition; working with major client developers

Data Models Q's

- How to enforce obscore consistency across archives?
    - Most columns are straightforward, some are "massagable" where necessary
    - obs_collection, calib_level, dataproduct_type, usage seems to vary between institutions; internally
- Improving mapping CAOM -> ObsCore
    - Resort to splat-ing the filename to capture file content; looking for more readable option
    - dataproduct_subtype, not obvious in CAOM

Datalink not just for ObsTAP/SIA

- Need to return products from Datalink API from multiple observations
    - input object_id, no id supplied,
    - object_id follows related product, resulting output ID has same ID=obs_publisher_did to allow useful grouping to clients

Questions:

- **Q: (MD)** Good point about datasource, marking simulated vs real data, SSAP has a way to declare this. datasource_metadata field with predefined vocabulary. Probably need to take SSAP method for registry to demarcate simulated data.
- **Q(FB):** For discovery of simulated data, Markus made a good point, we can probably define a standard_id for simulated data. re: Last slide; obsc_connection, always intended to be ? way to distinguish inside service for data collection. Consider a list of strings, given in self-description of the service by a service descriptor (in the case of SIA/DAP- for ObsTAP you can simply query the obs-collection field in advance to know the list) for future, imagine defining a vocabulary for this. For dataproduct_type, there is effort to define a new list; already online, among IVOA vocabularies. For example "dynamic spectrum" and "spectral cube" have been recently added in this vocabulary.
- **Q:** Within the ? we have similar issues, important to flag simulated data, we need to know that, naturally. Concerning grouping of flags; for us in High Energy; it's mandatory. We must group them.
- **Q(GDF):** Simulated data, might be in "real" or "alternative" universe; start with real data, then simulate what SPHEREx's data will be; for Rubin, started with completely simulated data. cases where mult datasets are in same "alternative" universe. Having some way to identify commonality, link to MD, is useful. Cuts across obscore, simulated images AND catalogs, whether the true input, or output of simulated data. Consider Theory WG? When you bring simulated up in firefly, appears over a REAL context image, not desirable.
- **GDF Comment in Zoom:**

> I think there's an additional issue around simulated data that's potentially of interest for IVOA standardization:

> Not only do we want to make sure users don't inadvertently take simulated data as real, there's also the separate problem that simulated data may be "in the real universe" where the simulation is based on real objects (this is what the SPHEREx simulator does) or they may be in a completely different simulated universe (this is what a lot of Rubin simulations do) and they may even, in specific cases, be in the same "alternate universe" with another specific simulation (as the current Rubin and Roman public simulation releases do).
>
> Being able to document this somehow, e.g., via a "universe ID", would help for a lot of things — for instance, it would help us display a simulated catalog query over a background context image from the appropriate universe.

How should ObsTAP+DataLink provide options to retrieve the same product from multiple locations (including cloud)? *(Tess Jaffe)*

As a bg:

> How do we offer multiple product retrieval options for DataLink? Had shelved it, but now needs to be done.

- Have been looking into preferred options, we have preference, looking for advice now.

0th order requirements:

- Don't want to break anything, and won't if it will, or others not interested
- ObsTAP / DataLink -- want to make sure this works the way they are intended
- When looking at rows in Obscore, already several dimensions of information,
- adding extra columns not preferred unless necessary.
  Requirements
- Serving data from multiple places, one for discovery, available in multiple locations, depending on geographic location, (mirrors), also on-prem and cloud
  - Client needs to see these options
- Get an ObsTAP/SxA/DAP table
  - End result, client sees default and alt ways to get data
  - Client makes a choice, sends that request, gets the data from where they want
  - Fornax use case, but also a generic problem, geo and data save locations
  - clients should need to pre-know data locations
    Example with Python
- Use a .get_data_options()
  - no specification, get the default, otherwise, get a link for that location
  - choice by the coder

    - does it belong it pyvo?
- Mocked in firefly,
    - data discovery in GUI,
    - given a choice when attempting to download
    - Doesn't make sense when working with data on server-- the server makes that choice
        - Users use this just to get the list of filenames-- then they do what they want
          How to communicate this client?
- Semantics, content-type; not ideal, meant to be HR, "#this-aws"? not meant for this
    - Using this, you get the same file, distinguished as being in multiple locations
- Service descriptors
    - Gives the location of data, call the datalink of choices in SD
    - Number of issues:
        - Putting in obscore:
            - Good: tells client how to find in batch request
            - Con: you might have multiple datalink services
        - Put it in datalink table itself
            - Has a #this, but also service descriptor, client can make choices,
                - Cons: several layers of calls
    - Back to adding a column
        - Want more than a URL; AWS + "us-east-1" + "paid/free" etc
        - Putting it in obstap result is a bad idea, what about to the datalink table?
            - Seems to be the best way; there are precedents; auth columns; SREgion
    - Others:
        - In the registry-- seems to be plainly bad idea.
            - User doesn't know at the start how/where to make the choice
        - Server decides-- can't do that for the client
          Bottom Line:
- Want options, give a vocabulary for them
- Datalink is flexible, many ways to do so.
- Need a solution soon.

Questions

- **Q: Pat Dowler:** Nailed the problem / complexity--- Datalink doesn't have a way of giving alternatives: 1-end-bunch of resources. Saying something is an alternate is not easy; using service descriptors is most conceptually correct; you get repeated calls to services; adds a lot of overhead, not great at large scales-- but roughly right approach dealing with alternative choices.
- **Q: Mark Taylor:** using local_semantics doesn't sound terrible; discussions on what to name that column; used for additional refinement of things-- not the best, not the worst.
- **Q: G.Landais:** We plan in VizieR(CDS) to provide datalink in Votable output (eg: SCS, and not only ObsCore table). VOTable will have an added column with a link to a datalink service. Are there simple way (other than adding an added Resource) to annotate the URL in a column to specify the content-type ?

- **Tess Jaffe:** which VOTable will have an added column, the one you get as a result of ObsTAP/SCS? Adding to ObsTAP was what I was told NOT to do. But I was told adding one to the DataLink result table is OK.
- **François Bonnarel (retired from CDS)**:
    - if among all the alternatives one should be the default I think the DataLink can be acccessed via a service descriptor in the discovery service (SIA, SSA, ObsTAP, or DAP) response. Then the access_url FIELD may contain the direct default access to the dataset
    - Inside the DataLink response, the description FIELD really can give the human reader required information to choose, but it's not useful for the software, sure.
    - adding a new FIELD is perfectly allowed but it will be mainly informative (like description) and will not be standard. hence it will not allow a generic client to use it to make some decisions
    - So I think "local_semantics" would be the best place to add the various access mode or location information. The FIELD is standard, but the content is to be defined by the data provider. Exposing this vocabulary can be done in an external document following the ivoa vocabulary rules.

Persistent TAP Uploads: A Prototype in DaCHS *(Markus Demleitner)*

What are persistent uploads?

- TAP supports uploading tables from beginning
- Tables are uploaded in request-- transient, requiring reuploads
- Give users the ability to put tables on server-side, across requests
  Prior Art
- CADC has youcat, 2018, an idea of PUT on VOSI-Tables
- ESAC has VOSpace uploads with TAP+, uses lots of authz.
  Basic Operation
- Upload: PUT-- in CADC's approach, goes to tables/table_name; here, to user_tables/table_name
    - Could be quite large; the idea is that /tables reflects all tables, including the ones you upload-- could be quite large
    - Therefore makes sense of a unique endpoints

DACHS prototype

- Fetch; GET -- on table_name
- and DELETE, naturally
  What to Return
- PUT returns an informative VOTable, possibly redirect to metadata table
- GET returns VOSITable
- DELETE-- another informative plaintext info
- Errors via DALI VOTables
  MD Retrieval
- GET on user_tables-- works for all user tables
  - in DaCHS, not supporting low detail request on tables
- If you anonymously upload a persistent table, and forget the table name-- out of luck
  CREATE Table
- ADQL extension; returns an empty table-- should it return something else? Probably a table, at least
  Discovery
- Capability rows-- could help declaring endpoints that are used
  - Mark Taylor suggested GET's against user_tables, with appropriate error HTTP status codes
  - wouldn't need update to TAP/TAPREgExt

Open Questions

- Extending destruction time (a la UWS)
- How do we create indexes? Not particularly portable between db systems
- Do we want user-user-shareable tables -- is it popular at ESAC?
  - **[Sara Nieto]** " Don't have exact numbers, but very much used in community."
  - **[MD]** "Unfortunate-- hope we could go without it, but I suppose we need an interface-- VOSpace"
- Scaling: DaCHS has no quota-- but in production definitely would need them-- how do people discover limits?

Questions

- **[gpdf]** Sharing user-created tables is also a requirement of the Rubin version of this. Sharing user-created tables is also a requirement of the Rubin version of this.
- **[gpdf]** Controllable sharing with other users is a requirement for the Rubin version of this service. We are evaluating YouCat at the moment.

Upload capabilities available in TAP+ services at ESA archives *(Jose Osinde Lopez ; remotely)*

Upload requirements at ESAC w/ TAP:

- two primary upload types / use cases
  - "On the fly" and upload endpoint
- On the fly -- From TAP 1.0:
  - Immediate query-specific data suport
  - Uplaod data, run the query, delete data in server immediately
  - Well supported through many clients, python, etc
  - Using a job result in a query:
    - Upload local data to a TAP service, run a query against job executed, etc from TAP 1.0
    - Data is uploaded to DB as a new table during execution, use a different schema for uploaded tables for further ADQL queries, naturally
- Persistent Uploads:
  - Implemented at ESA:
    - Only for authenticated users, you already have to have a DB of users,
      - One of the problems: if there is an anonymous user, unknown how much to care (essentially) about their uploaded table, and for how long?
    - No table name conflicts this way, username in schema id
    - Data is private per-user; how do we then share that table with other users? Still needs implementing
    - Quotas: data needs to be checked on every upload, check the user quota
  - Upload endpoints
    - uses /Upload endpoint
    - Deletions are performed by a POST with a DELETE=TRUE parameter, optionally force_removal, to force deletion of shared table across all users
  - Sharing endpoint,
    - Provide a list of resources,
    - Administration of list of users/groups, assign the resources to the groups
    - Handled with GET requests with parameters for creating group, adding users, creating shared item, then a link between shared items to users
  - Upload is required for ESA, but complex
  - Many endpoints, with many parameters-- complicated but doable, requires a conversation, ESA is open to modifications

User-managed tables in TAP *(Pat Dowler)*

Use cases for user-created tables:

- TAP 1.x already has temporary tables, good for small ones, but not larger, using UPLOAD
- upload of temporary for use in multiple queries
- upload a private table, permanent;
    - great for sharing between users,
    - depends on auth and group membership administration
    - -- and can publish it for later

Core functionalities

- Implemented in youcat using the VOSITables endpoint -- PUT, DELETE and POST added for create, drop, update respectively
    - adding rows: VOTable could be used for small tables,
    - you can create a table definition by querying another table-- or tap service-- and copying
    - For large tables, users can append rows to existing tables, for scalability-- you can't put very large tables up in one go
- VOSITables spec can be updated to support new operations
    - most services would only have basic support
    - some provide the complete tables
    - OpenAPI / P3T: how to make endpoints optional?

Necessary functionalities

- Create / drop index
    - async UWS job in youcat
- Grant / revoke permissions
    - allocation is a schema owned by a user-- implemented an extension of TAP_SCHEMA, additional user, anon read, read/write groups all store in the TAP_SCHEMA
        - mostly for project collab / sharing
- Necessary (probably) to update tap to specify index jobs, custom endpoint for setting permissions (optional)

Stretch goals

- Allow users to attach service descriptors to columns -- exploratory work, report backa t next interop
- Publishing tables: make table visible to registry search
    - Because they're already in vositables, etc. existing clients could find them
    - A lot of QA would be required for table metadata; probably the responsibility of the service implementor
        - Updates to user-uploaded table metadata would need to be handled

        - As reg moves towards findable tables, probably desireable, but a considerable amount of work
    - Implementation specific, but not required: users can add/improve metadata, but won't require them too

Questions

- **[gpdf]** Rubin would be very interested in talking to CADC about the service-descriptors-on-user-tables thing.

Questions on uploads

- **[MT]** TOPCat will find user-uploaded tables--- be consistent in what is registered. We shouldn't standardize something that does this kind of publication, curation should be encouraged. A difference between each implementation, these uploaded tables DO NOT appear in VOSITables/TAP_SCHEMA. I would claim for these uploads, that supports POSTing, would know where to fetch the metadata already, makes the service implementation easier. Users won't need to discard tables from VOSI Tables
    - **[?]** Possible to segregate standard and user tables; a viable straregy, the two endpoints would have same api's. Doesn't mean that until a client is updated to know there's no use.
    - **[?]** Don't know how we can implement this segregation to be implementor's choice. Aiming for sharing in project as primary use case, api parts are straightforward.
- **[?]** If you put user table in registry, will/how we separate users from institutions? We should add some flag, etc to distinguish
    - **[?]** Yes, I agree.
- **[gpdf]** Agree on the need to flag user tables in any registry records that may be created
- **[?]** If you have two different kind of tables,
- **[?]** DIfferent schema means it's still visible to VOSITables? (Yes) You have VOSpace implementation, seems reasonable to use VOSpace to do this-- I'm glad you didn't. Was this your only consideration?
    - **[?]** They upload directly to database, not to VOSpace.
- **[MD]** What is the UI/API people use to upload?
    - **[Jose Osinde]** We also have a VOSpace, for GAIA we have our own spaces in the archive database. These user spaces are typically used to run cross-match queries against a catalog in the archive or for validation purposes before data is released, for instance.
    - **[PD]** I think there's is similar to ours, in our system auth is necessary-- quotas are a necessary evil when it comes to users using server resources. Quotas not currently implemented, we simply nag user since they're already authed. We tried to use VOSpace-- you could upload them, then tag with notes you want them in TAP-- that was in VOSpace because you can add optional things--- it's really not an API, don't let you specify exactly how that works, you'd need a bunch of ad-hoc properties, that would all be async, communication of error messages when things go wrong is completely undefined. Seemed to be a more complex way to do the same thing, just to save on authorization, which only requires 4 values.
- **[MD]** How did you solve these? Now that you've used vospace

- **[Jose Osinde]** I agree, it'--- you want to have --- in our case, don't do manual monitorting of quotas, a lot of effort to make sure the system is working correctly. --- So we enforce the quotas in the system from the beginning. This also involves a management tool that associates a given quota with a user, monitors all actions that affect the quota space (used or assigned), and provides appropriate feedback to the user, including notifications of any quota violations.
- **[GDF]** For rubin, the youcat approach is a good match to requirements we've had. it's essential we have VOSITables/TAP_SCHEMA MD. Users should be able to declare joinable user tables to system tables. We need auth control capabilities, so people can share with groups or larger community. Publishing-- in the long term (doi's), not something we've tackled, perhaps lifetime of a year so they can be shared with collaborators. Haven't evaluated TAP+ closely, could work just for the options presented. Feel strongly that having exst. clients work with these user tables, important and positive feature.

Below topics cut for time reasons.

Cone Search 1.1 - Status *(Marco Molinaro)*

ADQL-2.2 - PEG grammar - Status *(Grégory Mantelet)*