IVOA Data Access Layer
Table Access Protocol Analysis

Doug Tody (NRAO/NVO)

TAP Design Study

- History
  - Based upon work done by ESAC/VOQL-TEG and DAL WG in spring 2007
  - Also NVO tiger team, SkyNode experience, data center experience
- TAP Design Goals
  - Provide capability for ADQL queries to support advanced analysis
  - Define minimal implementation
    - for small data provider, common queries
    - replace legacy cone search with more general facility
  - Both data access and metadata access supported natively by service
  - Provide for scalability, in particular multi-position queries
  - Support Grid capabilities, i.e, async, staging, authentication
  - TAP should be consistent with other DAL interfaces where possible
  - Provide registry integration for automated service discovery

TAP Interface Summary

- Form of interface
  - HTTP GET/POST based (other protocols possible, e.g. SOAP, CEA)
  - Multiple output formats (VOTable, CSV/TSV, XML, VOSpace, etc.)

- Operations

– AdqlQuery	ADQL-based queries, full functionality
– SimpleQuery	Simple data queries, metadata queries
– GetCapabilities	Return metadata describing the service
– GetAvailability	Monitor runtime service function and health

AdqlQuery Operation

- Scope and Form of Interface
  - General capability for ADQL-based queries
  - Both GET and POST versions are required
    - GET is synchronous, indempotent, simple, RESTful
    - POST required for async, staging, large queries
  - Semantics, e.g., parameters, identical for both versions
  - ADQL query is URL-encoded so use in GET is not a problem

- Parameters

– QUERY	The query string (ADQL; URL-encoded)
– FORMAT	Output data format (VOTable, CSV, XML, etc.)
– *<staging>*	Only used in POST version; for VOSpace
– *<async>*	Only used in POST version; for driving UWS
– MAXREC	Maximum records in the output table
– RUNID	Pass-through; used for logging

(others TBD)

AdqlQuery Operation

- Field Names, UTYPE and UCD
  - Suggest this be done at level of field rather than by operation
  - Literal field names directly access database table
  - A UTYPE reference resolves into a literal table field name
    - e.g., “ssa:Target.Name” resolves to table field “TargetName”
  - UTYPE (in this context) is a special case of UTYPE ("ucd:")

- Field name resolution
  - Both literal and UTYPE/UCD field names resolve to table field
  - All queries evaluated equivalently after field name resolution
  - Data models, at the level of TAP, involve only mappings
  - UFI can automate this, or it can be done client side

AdqlQuery Operation

- Multi-Position Queries
  - AKA multi-cone search; but doesn't have to be limited to position
  - Common use-case involves user source list with thousands of positions
  - Required for scalability to reduce operation overhead
- How It Works
  - Uses ADQL, REGION, POST form of operation
  - VOTable used to upload source table (ID, POS, SIZE, etc.)
    - other fields are passed through to output
    - output is tagged by source ID
    - can be generalized to any input parameter, not just position
  - POST (e.g., multipart/form-data) used to upload params, VOTable
  - Parameters are common to both GET and POST forms
- Data Scoping
  - Query, Local (DBMS), and VOSpace (Net) tables are equivalent
  - POST is a Query space table

SimpleQuery Operation

- Scope and Form of Interface
  - Provides capability for simple non-ADQL queries
  - Used for both data queries and metadata queries (like ADQL/SQL)
  - Only a synchronous GET version is required
  - Only a single table is queried at a time
- Motivation
  - Simple to implement, easy to use
  - >90% of actual catalog queries are simple filters of a single table
  - We need something like this anyway for simple *metadata* queries
    - but why limit it to only metadata?
  - Small data providers publish a few simple catalogs
  - Simpler to implement, likely to be more robust implementation

SimpleQuery Operation

- Parameters

– SELECT	Table fields to be returned (default all)
– FROM	The table (or view) to be accessed
– WHERE	A filter to be applied to the table (default none)
– POS,SIZE	Find data only in this spatial region
– FORMAT	Output data format
– MAXREC	Maximum records out
– RUNID	Pass-through for logging

  (other params TBD)

- Provides
  - Simplified SQL-lite query (90/10 rule)
  - Both data and metadata queries
  - Simple cone search capability

SimpleQuery Operation

- Metadata Queries
  - Information Schema concept
    - great concept; definition/implementation imperfect
    - but it is a standard, widely (but not completely) implemented
  - Concept
    - represent database/table metadata as data tables (views)
    - allows use of standard data table interface to query metadata
    - easily extensible without changing service interface
    - views can be used for things such as registry view
  - Examples
    - FROM=SCHEMA.tables
    - FROM=SCHEMA.columns&WHERE=tableName,foo
    - FROM=SCHEMA.columns&WHERE=tableName,foo&FORMAT=xml

Simple Cone Search

- Approach
  - Integrate into SimpleQuery to allow additional constraints
    - would probably be too ambitious in a separate SCS standard
  - Re-use common DAL position syntax (POS, SIZE)
    - extensible in terms of region type and spatial frame
  - UTYPE/UCD field syntax allows data models to be used
  - Table to be queried is specified with FROM
  - ADQL,REGION provides an advanced alternative with common semantics

- Examples
  - REQUEST=SimpleQuery&FROM=foo&POS=180.0,12.5&SIZE=0.2
  - REQUEST=SimpleQuery&FROM=foo&POS=180.0,12.5&SIZE=0.2&WHERE=flux,5/

Minimal TAP Service

- Requirements
  - Implements SimpleQuery operation
    - possibly getCapabilities and getAvailability as well?
  - Provides basic data query capability
  - Provides basic metadata query capability (tables, columns)
  - No ADQL support required (but may use SQL back end)
  - No UTYPE support required