Astronomy and Computing 7–8 (2014) 37–44

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

The Table Access Protocol: Providing standard access to astronomical data

D. Nandrekar-Heinis [a,*], L. Michel [b], M. Louys [b,c], F. Bonnarel [b]

[a] Johns Hopkins University, 3400 N Charles Street, Baltimore, MD 21218, USA

[b] Observatoire astronomique de Strasbourg, Université de Strasbourg, CNRS, UMR 7550, 11 rue de lÚniversit e, F-67000 Strasbourg, France

[c] ICube, Université de Strasbourg, CNRS UMR 7357, 300 bd Sébastien Brant - F-67412 Illkirch Cedex, France

ARTICLE INFO

Article history:
Received 5 May 2014
Received in revised form
11 August 2014
Accepted 13 August 2014
Available online 23 August 2014

Keywords:
Table Access Protocol
Virtual observatory tools
Virtual observatory standards
Astronomical databases: miscellaneous
Catalogs
Surveys

ABSTRACT

In the upcoming era of large scale, geographically distributed, varied sources of astronomical data, a standard, simple and flexible way to access this data is necessary and useful for astronomers across the globe. Most of the modern surveys such as the Sloan Digital Sky Survey (SDSS) are available in well organized tabular formats. The Table Access Protocol (TAP) supports a standard web interface to access this kind of tabular data. Predefined queries and results formats help different data providers to implement these services. TAP also helps various software tools to access data and perform cross-matches seamlessly across different data sources. It is then possible to access data in tools that consume TAP web services. This supports further detailed data analysis on a queried slice of data. This document describes TAP and its utility for astronomers and data scientists. It also provides information on the protocol for data providers and developers.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Over the last decade, astronomy has seen a true data deluge, helping astronomers to put more constraint on physical processes and open new research opportunities. However, this comes with the price that astronomy data is heterogeneous in every sense of the word. The data has been taken by different instruments, in different wavelengths. The datatypes (images, spectra etc.) differ as much as the units (magnitude vs. fluxes). On top of this, the data is spread all over the world, and stored in various ways either in well-defined relational database management systems (RDBMS) or on file system as raw files (using different formats as well). Astronomers usually need to gather and combine all the data in order to take full advantage of the information available. This means for instance cross matching sources by position to study their properties including colors or other physical parameters. The heterogeneous nature of the data makes these tasks a challenge for astronomers. The astronomy community then recognized that there is a need of data standardization and homogenization. In 2002 the International Virtual Observatory Alliance (IVOA, http://www.ivoa.net) was formed to create an environment of interoperable datasets, resources and services. It creates standards and protocols for data storage to data access via Internet. Standardization is very important for interoperability. Various standard Internet protocols were used as a basis for the IVOA protocols, the main examples being HTTP and XML. The IVOA designed and proposed a number of WebService protocols to access different kinds of data resources, such as the Simple Cone Search Protocol (SCSP), Simple Image Access Protocol (SIAP), and Simple Spectral Access Protocol (SSAP). According to IVOA these protocols are called 'simple' because they are dedicated to one type of data and have a limited query parameter interface. Moreover, they provide specialized operations appropriate to each dedicated data type, such as "cutouts" for images or pixel transformations (SIA1). The emergence of these web services also led to the development of various client tools with simple, user friendly, interfaces for end users, including TOPCAT (Taylor, 2014), Aladin (Bonnarel et al., 2000) (see Appendix B) etc. But the development of these Virtual Observatory (VO) clients along with some specialized clients also pointed out that "Simple" protocols do not allow heterogeneous data access, offer only restricted querying capabilities, and have limited data description capabilities. The solution to these issues is the more flexible protocol called the Table Access Protocol (TAP, Dowler et al., 2011), started in early 2007. TAP gives flexible yet standard access to different kinds of databases and helps to unify client development

* Corresponding author. Tel.: +1 4105164102.
E-mail address: deoyani@pha.jhu.edu (D. Nandrekar-Heinis).

http://dx.doi.org/10.1016/j.ascom.2014.08.002
2213-1337/© 2014 Elsevier B.V. All rights reserved.

as well. TAP is based on the Representational State Transfer (REST). Using a TAP service user can query any database and get results, upload some data and perform operation on server, create temporary table of their own data on a server. These data tables can keep their original columns and do not need to be standardized. TAP by default accepts query in Astronomical Data Query Language (ADQL, Plante, 2007), which is an IVOA standard based on Structured Query Language (SQL). Astronomy users have their own simulated data or data slices from different sources which they want to match with large surveys. In this case it is better to upload the data on server which hosts large data and perform cross match there. Astronomy data users and software developers may prefer to write their own scripts/applications to perform data analysis or to use existing tool. In either case they need to develop and integrate a VO layer to be able to interface with the standard services and to interpret the standard formats. TAP proves to be helpful in this case because of its flexible nature.

TAP in VO Context:

The VO Data Access Layer (DAL, Dowler et al., 2014) provides protocols that standardize access to data. For example, VOTable (Ochsenbein et al., 2011), is based on XML and can contain both metadata and result data in one single well structured document. These are not dependent on input types and they are easy to consume by various clients. TAP supports these standards.

The following example illustrates a typical VO sequence of data discovery. The VO protocols make a distinction between tabular and non-tabular data. Tabular data consist of keyword values extracted from data files and stored in tables (one column per keyword). Non-tabular data is typically of the form of N-dimensional data arrays (spectra, images, data cubes).

1. A user needs a simple way to find the service which deliver VO-Compatible data. This is done by querying VO registries, which provide description of types, availabilities and capabilities of VO services as well as URL of these services.
2. After choosing the service(s) a user is now able to send appropriate queries to them. Tabular data are queried using ADQL, and in most cases the responses are list of measurements within the application or downloadable for further research. Thus, querying scientific tabular data with TAP in the VO is generally a two-step process: service discovery followed by data discovery and retrieval.
3. Non tabular arrays of values can be discovered using ObsTAP (Observational Table Access Protocol, Louys et al., 2011, for details see Section 2.5) or "simple" protocols.
   (a) ObsTAP is a TAP service that can be queried using ADQL, where the response is a description of non-tabular datasets. ObsTAP is used to retrieve metadata of datasets (descriptions of data consistent with one of the IVOA data models, namely ObsCore). The application will display metadata embedded in the query response of these services in its interface, thus providing a dataset discovery facility. The user can then download the full datasets.
   (b) "Simple" protocol queries provide basically the same capabilities, but the Query Parameter interface is less advanced there. The specialized protocols also allow direct access to data subsets without full data retrieval.
   (c) Thus, access to non-tabular data in the VO is typically a three-step process: service discovery, followed by data discovery and description, and then data retrieval.

The outline of this paper is as follows: in Section 2, the details of the TAP standard are described. In Section 3, we describe how to implement TAP; this section is intended for data providers and developers who wish to serve data using TAP and to write TAP clients. We also provide additional implementation notes in Section 4. Users who want to explore how to do science using TAP are advised to read Section 5. Currently available tools and clients are listed in Appendices A and B. We discuss pros and cons of TAP in Section 6. Future developments are outlined in Section 7.

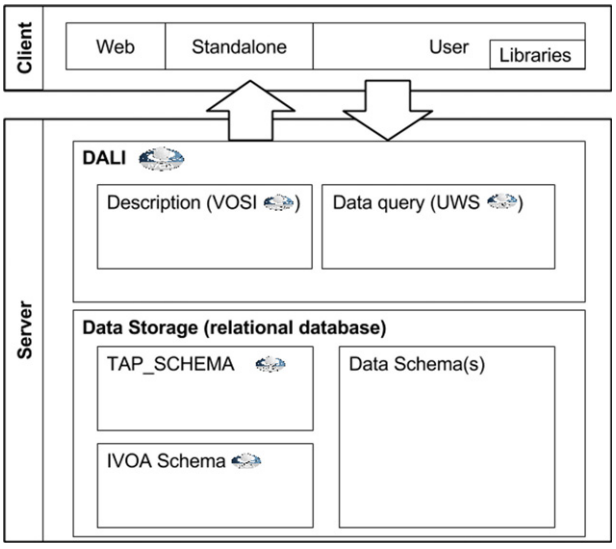

Fig. 1. VO building blocks supporting TAP client–server interactions. Here are the building blocks that are used for searching, locating, transporting tabular information between client end API (top layer) and server end with data bases (bottom layer). The protocol relies on the DALI interface standard to describe all the service capacities, and on UWS, for the definition of the asynchronous web service adopted in the VO. The VOTable tabular format is used to store and circulate all data or metadata. TAP schema specifies the content of these tables. TAP can transfer any content described by the TAP_SCHEMA definitions. Datasets compliant to some specific data model, like ObsCore for instance, can be exposed with the dedicated IVOA ObsCore *TAP_SCHEMA*. ObsTAP is a specific TAP_SCHEMA that defines columns for the data model items of ObsCore DM.

2. Description of the Table Access Protocol

The TAP protocol can serve any kind of tabular data, image descriptions, source catalogs, etc. It is a standard way to expose either a portion of or a complete astronomical relational database into the VO. Fig. 1 shows the architecture of a TAP service. The database interface is compliant with the Data Access Layer Interface (DALI, Dowler et al., 2014) specification. DALI defines resources, parameters, and responses common to all DAL services. As such, a TAP service is a RESTful, web service supporting a standard set of resources (endpoints). Some of them are in charge of the query processing Universal Worker System (UWS) and the others compliant with the Virtual Observatory Support Interface (VOSI). TAP puts very few constraints on the database organization. It just requires the presence of a particular schema describing the exposed tables, the TAP_SCHEMA. Two other schema names are reserved (1) IVOA to store data tables compliant to a VO data model and (2) TAP_UPLOAD which is a transient schema containing the tables uploaded by the users. The regular data tables can be packaged in the schemas as per the database design. However there is no requirement on the relational database to support these schemas; they can be emulated by the service (see Table 1).

Any other data storage system can be used behind a TAP service. The default TAP query language is ADQL and the default output format is VOTable.

2.1. The service description

A TAP service must implement three resources describing the supported capabilities, the published content and the service availability. These resources are compliant with the VOSI standard which specifies the schema applied to the XML responses.

REST architecture helps accessing the description of a TAP service with a simple browser. The `/tables` resource is very useful for services like TAP, which serves heterogeneous data. It provides a metadata description of tables containing data to be queried to

Table 1
DALI resources.

Resource type	Resource name	Description
VOSI-availability	/availability	Check status of the service
VOSI-capabilities	/capabilities	Description of the supported capabilities. (e.g. query languages, output formats, limits for execution duration, response size, etc.)
VOSI-tables	/tables	Description of published schemas and tables

Table 2
TAP schema.

Table name	Content
Schemas	List of exposed schemas
Tables	List of exposed tables
Columns	List of all columns (name, data type, UCD, utypes)
Keys	List of foreign key relations between tables
Keys_columns	List of columns part of a foreign key relation

clients. Accessing this metadata is essential for the user or client to write queries. The particular service capabilities can also be determined by queries to the VO registry.

2.2. The TAP schema

The subset of the underlying database exposed in a TAP service is described in five tables gathered in a schema named TAP_SCHEMA (see Table 2). These tables contain a description of the schemas (the tables and the table columns). This description is made of both metadata (name, data type, and units) and a few semantic labels such a text description, a Unified Content Descriptor (UCD) which is a semantic class to the described quantity, or a UTYPE pointing to a data model node. The TAP_SCHEMA also contains a description of how tables can be joined. These table joins help users to retrieve the components of complex data split over several tables. That is very useful for data organization based on a hierarchical model. The existence of declared joins does not prevent the user to join tables in another way.

It is important to note that in a TAP service, the metadata is stored in data tables, which means at the same level as the data. Thus, it can be explored with the same tools as for data retrieval (ADQL queries). The contents of the `/table`, a VOSI resource, can be retrieved by querying the TAP_SCHEMA.

2.3. ADQL

ADQL is the default query language for TAP. It includes most features of SQL. There is no limitation on the way to filter data (WHERE clause) nor any restriction on the columns of the query results (SELECT clause). That makes TAP queries more flexible than those of the "Simple" protocols where the set of returned columns is fixed by the protocol. In addition, ADQL supports spatial search functions making positional cross-matching between tables easy. ADQL can also access temporary tables uploaded by the user. ADQL does not support embedded procedures or statements modifying the database content.

Here is an example of an ADQL query for positions of the United States Naval Observatory (USNO) sources located at less than one arc minute of the center of Messier 1 (data source: German Astrophysical Virtual Observatory (GAVO) data center).

```
SELECT d.raj2000, d.dej2000
FROM usnob.data as d
WHERE  CONTAINS
(POINT('ICRS', d.raj2000,  d.dej2000),
CIRCLE('ICRS', 83.633042, +22.014500, 0.0166))=1
```

On the client side, connecting a TAP service is quite easy since the protocol is RESTful. There is one URL per resource which returns either XML files (job status or metadata) or VOTable files (error reports or data). Other response formats, such as FITS, ASCII tables may also be supported.

2.4. The query processing

A TAP service can process queries in both synchronous and asynchronous modes. These capabilities are part of the DALI specification (Table 3). The choice of mode depends on the client. Basically, queries liable to take a long time to process (more than 1 min typically) should run in asynchronous mode.

Synchronous queries are stateless. The HTTP request parameters (query string, output format etc.) are passed either in GET or POST HTTP mode. The request returns the result or an error report, in a VOTable or any other supported output format. Asynchronous queries are processed by stateful requests based on the UWS pattern. The request parameters are passed in POST mode. The response is an XML file containing informations about the jobs and an identifier (jobid) used to address further operations. The client is in charge of monitoring the job phase before the transfer of the result after completion. The retention duration of the job is defined by the service capability.

If the execution duration exceeds the limit imposed by the service, an error status is enclosed in the resulting VOTable. If the number of rows exceeds the maximum allowed by the service, the result is truncated and flagged.

2.5. Data models with TAP: the case of ObsTAP

ObsTAP provides a general discovery protocol for any kind of data product gathered and distributed in the VO: spectra, sky images, light curves, event lists, radio or hyper-spectral cubes, etc. ObsTAP is based on ObsCore data model (Louys et al., 2011). ObsCore defines the basic set of metadata, necessary to describe search criteria to discover and access observations across various archives and data centers. This model deals with three main metadata categories:

1. The identification of a dataset, its data product type, collection and curation details.
2. The data format and access parameters.
3. The characterization of the physical axes represented in the dataset. This deals with location, ranges, resolution, sampling features estimated for the physical measurements involved in the dataset: spatial, spectral, temporal, flux, polarimetry, etc.

From this common physical description, search criteria like reference sky position, covered sky region, spectral interval, time interval, polarimetric type, etc., can be formalized with queries valid for different data archives supporting this protocol. This model is specified as a structured set of classes and their attributes. ObsTAP is the representation of these data model items in the TAP framework. A TAP service can only host one data table for ObsTAP. This table is named ObsCore and must be stored in the IVOA schema.

3. TAP in action

3.1. Data provider: how to publish one's data with TAP

The deployment of a TAP service is done on the top of an existing database. This includes several tasks:

Table 3
Jobs processing endpoints.

Resource type	Resource name	Description
DALI-async	/async	Run asynchronous queries compliant with the UWS pattern.
DALI-sync	/sync	Run synchronous queries

Table 4
Binding the TAP_SCHEMA metadata with the selected columns.

Query	Selected column	Unit	Location
`SELECT s_ra FROM ivoa.obscore`	pos.eq.ra	deg	Taken from the TAP schema
`SELECT floor(s_ra) FROM ivoa.obscore`	not set	deg	Inferred from the TAP schema
`SELECT radians(s_ra) FROM ivoa.obscore`	not set	rad	Inferred from the TAP schema

1. Construction of the TAP_SCHEMA: it can be achieved by a couple of SQL queries. The tables which are not referenced by the TAP schema are not exposed by the TAP service. This means that they cannot be discovered by browsing the TAP schema but they might however be accessed by an ADQL query. Completely hiding data tables requires either to use the permission system of the RDBMS or to do a pre-checking of the query (or both). This issue is out of the scope of the TAP protocol. It is also possible to expose SQL views of real tables. This allows to hide columns or to include implicit unit conversions or implicit joins.
2. Building the ADQL to SQL translator: Although being close to SQL, the ADQL translator requires a parser to work properly. The ADQL BNF grammar is provided in the ADQL specifications. The translator must first take into account the underlying RDBMS by taking care of reserved words (e.g. "keys" is both a reserved name in MySQL and the name of a table of the TAP_SCHEMA) and restoring the proper character case (e.g. MySQL is case sensitive, ADQL is not). Finally the spatial functions must be implemented.
3. Building the module formatting the QUERY result into VOTables (and other supported formats).
4. Building the REST interface handling the synchronous capabilities.
5. Building the REST interface handling the asynchronous queries (UWS resources). This task can be difficult since it must manage complex features such as user sessions, job queues or jobs timeouts.
6. (optional) Implementing the upload feature.
7. (optional) Implementing the ObsCore view.

Various tools and toolkits that help perform the above tasks are listed in Appendix A.

3.2. Data access: how to write a TAP client

Since TAP is a RESTful protocol, writing a basic client does not need any detailed programming. The client should process URLs and parse VOTables. In fact user familiar with web services will find it very simple using command line programs like CURL. Some users however, need some help to locate and select TAP services. This can be achieved by browsing registry and selecting desired TAP service. If users do not want to write their own ADQL query, client tool can provide simpler interface to achieve it and query on behalf of the user. The same client can be used to display results in the user friendly interface. Following are the steps to write a TAP client.

1. Use VO Registry to locate available TAP services.
2. Give access to users to select desired TAP service or specify the TAP service URL.
3. Create UI for user to specify different parameters for query.
4. Create interface to build SQL query based on user criteria.
5. Option to submit query to the selected TAP service.
6. Show results returned by the TAP service in user preferred format.
7. Provide an option to download these results.

Available TAP clients are listed in Appendix B.

4. Implementation notes

The most notable feature of TAP is the flexibility, which results from the use of a structured query language. This feature must be taken into account at all stages of the development of TAP service or client. Below are a few situations that require special attention while implementing TAP.

1. Resource Intensive Queries: To take care of queries using too many resources is a bit of problem in SQL Server. This problem arises because it is impossible to predict the duration of query execution or to compute the resources consumed by a particular query. It is for instance easy to ask for a full cross match of two very large catalogs simply by omitting or mistyping a selection filter. The protocol requires to put a limit on that time, but the implementer must pay attention so that aborted jobs actually free the database resources (CPU + I/O). Query parsers should also be well designed to catch ambiguous queries.
2. Response metadata: It is important to make the metadata of a response compliant with the data. The columns of the query responses do not necessarily match the columns of the data tables as described in the TAP_SCHEMA, as they can result from computations, e.g.,

```
select (t-min + t_max)/2 as t from tableName
```

or they can be altered by some function (see below). In some cases, that can make the binding of the response columns (SELECT items) with the metadata a bit tricky. Table 4 shows a typical case illustrating this issue. In this example the parser should be able to know the function applied to the field `s_ra` (floor, radian) to properly set the metadata in the response. In particular, the service should remove UCDs and change units. This kind of fine tuning is usually not implemented. The end user is supposed to interpret the results properly.
3. Querying complex data. On the client side it can be difficult to state queries when the model of the searched data is very intricate and/or hierarchical because TAP just exposes a relational view on that model. The relational implementation of the registry (17 tables)[1,2] provides a good example of this. The query below selects all SIA services that mention "spiral" in either the subject, the description, or the title of the resource.

```
SELECT ivoid, access_url
FROM rr.capability
  NATURAL JOIN rr.resource
  NATURAL JOIN rr.interface
  NATURAL JOIN rr.res_subject
WHERE standard_id='ivo://ivoa.net/std/sia'
  AND intf_type='vs:paramhttp'
```

[1] http://www.ivoa.net/documents/RegTAP/.
[2] http://code.google.com/p/volute/source/browse/trunk/projects/registry/regtap/RegTAP.html.

```
AND (
1=ivo_nocasematch(res_subject, 'spiral')
OR
1=ivo_hasword(res_description, 'spiral')
OR
1=ivo_hasword(res_title, 'spiral'))
```

Building this query from scratch is clearly not easy. Extra knowledge of the model is necessary for the user or for the client developer along with the metadata delivered by specified TAP service. This knowledge can be hard-coded in the client (e.g., query template or simple form), but it can also be inferred from the metadata describing the service (the `/tables` endpoint). This example shows the necessity of making metadata as complete as possible (type, description, range of values) to make the best use of a given TAP service.

4. Extensive metadata. The current version of the TAP protocol is not suitable for resources containing a large number of tables because all metadata is returned in one response. Vizier,[3] for instance, exposes more than 100,000 columns shared among 16,000 tables (CDS, private communication). The corresponding XML file cannot be ingested by most parsers. The Vizier TAP service already diverts from the standard by delivering metadata table by table. The same issue can arise with simulation data, which can have tables with a very large number of columns.
5. Workflow consistency. People implementing workflows including TAP search must be aware that the format of a TAP response, the returned columns, is set by the query and not by the protocol. That might be an issue for the definition of task interfaces.
6. Accessing non-tabular data. The TAP protocol only gives access to the data stored within the exposed tables. There is no way to access image pixels without downloading the whole file, assuming that the download link can be retrieved within the response. This limitation will be overcome by emerging protocols such as SIAV2[4] or DataLink,[5] which allow invocation of an external service, returning the data attached to one specific row (see Section 7).

5. Science with TAP

One of the main benefits of TAP lies in the ability to query a service without preliminary knowledge about both data content and data schema owing to the TAP_SCHEMA and the VOSI interface.

Use Case 1: Build a photometric redshift training set from SDSS photometric data. This is a simple example using one TAP service.

Approach:

1. Get photometric data.
2. Identify spectroscopic data from the relevant database.
3. Match the two datasets on a given condition to check which photometric objects have spectroscopic redshifts.

Using TAP:

1. Search the SDSS catalog's PhotoObj table for the objects or desired area of sky with unique object ids.
2. Search the SDSS catalog's SpecObj table for spectral information on the similar objects. Write the query to join these two datasets on a particular Object Id (objid = bestobjid).

Use Case 2: Upload user data to a TAP service and join with existing data in a given TAP service. For example, what is the correlation between optical and far-infrared properties of clusters of galaxies?

Approach:

1. Look at the infrared properties of galaxy in a given cluster. Get the infrared data.
2. Get the optical data for the corresponding cluster of galaxies.
3. Do the angular cross match on the two datasets.

Using TAP:

1. Identify the services that provide SDSS and WISE data.
2. Connect to WISE TAP service and download the required data by querying the service.
3. Connect to SDSS TAP service; write the proper query which will perform the angular cross-match. Upload the relevant WISE data, and run the query (e.g., search SDSS sources within 3 arcsec of WISE detections).
4. After running query download the results. Plot correlations (e.g., optical colors vs. far-infrared luminosity).

Use Case 3: Get all the XMM-Newton scientific products (images, spectra, time series, etc.) related to one target detected by the spacecraft and referenced in the Third Catalog (namely 3XMMdr4) published by the Science Survey Consortium.[6]

Approach:

1. Avoid picking data in multiple tables or resources.
2. Locate the XCatDB (Motch et al., 2009), one of the interfaces of the XMM-Newton catalogue.
3. Collect scientific products gathered in an ObsTAP resource from Table 4.

Using ObsTAP:

1. Locate the TAP interface of the catalog having an ObsTAP capability.
2. Select the ObsCore table.
3. Get all datasets located around the target.
4. The response contains pointers to the data files with comprehensive metadata giving the boundaries in the space, time and energy axes.

6. Discussion

The strength of TAP is clearly its general and reusable template for accessing any kind of database content. It does not depend on the RDBMS system used, nor on the programming language used on either the client or server side. ObsTAP expands the possibility for wide range data discovery of observation data products across diverse data archives. This is due to the generality of the ObsCore data model, which gathers concise metadata descriptions for selecting a data product of interest.

The flat tabular nature of this protocol is simple enough to cope with many different kinds of data and metadata, as illustrated in the various implementations in Appendix A.

The tables described in the TAP_SCHEMA can be extended with new columns, therefore evolving, upgrading a TAP service can be achieved with only minor re-programming. For databases containing data both interrelated and hierarchical, TAP Schema needs many tables to represent the relations inside the hierarchy. Exploring such data collections induces many `join` operations.

This sort of query stated with ADQL can have an unpredictable execution time. TAP may fail to warrant effective searches in reasonable time. In these cases, dedicated services with predefined search criteria like SSA (Doug et al., 2012), SIAv2, are able to focus on optimized criteria and discover the data of interest in more efficient way.

The `/upload` facility is a key point to facilitate comparison on large amounts of data, including of course position cross-match. It is also useful to build-up reference datasets by extracting VO data from multiple data centers, and scoring some specific parameter, like for instance spatial resolution, detection limit, etc.

[3] http://vizier.u-strasbg.fr.

[4] http://www.ivoa.net/documents/SIA/.

[5] http://www.ivoa.net/documents/DataLink/index.html.

[6] http://xmmssc-www.star.le.ac.uk/.

Table A.1
Publication tools.

Publication tools	Description
Saada http://saada.unistra.fr Strasbourg Observatory (Fr)	Integrated tool including the database creation and management, a Web interface and the VO services (Michel et al., 2006). Use the CDS/ARI library.
DaCHS http://vo.ari.uni-heidelberg.de/soft/dachs GAVO (Ge)	DaCHS is a multi-protocol toolkit containing a full TAP stack including geometries and upload.
OPENCADC https://code.google.com/p/opencadc/ CADC (Ca)	Java toolkit for deploying a TAP service. Open-source branch of the software developed by the CADC for its services.
VODance http://ia2.oats.inaf.it/index.php/vodance INAF (It)	VODance is a tool for rapid Virtual Observatory compliant services deployment.
CDS library https://github.com/gmantele/taplib CDS (Fr) + ARI (Ge)	A pack of 3 Java libraries (ADQL, UWS and TAP) helping to build a TAP layer on the top of an existing database.
TAPtoolkit https://sites.google.com/site/usvirtualobservatory/home/tapserver VAO (US)	To be released (it is to help data provider their own data without coding all TAP service themselves, it helps as a guideline).

Here is a list of tools provided by the IVOA to the community for publishing data sets in the TAP framework. These are meant to provide a TAP layer to a data archive.

7. Summary and future developments

The success of TAP stimulated further developments of various toolkits taking advantage of cloud technology and facilitating the scientific exploitation of user data in a TAP context. Appendix A lists various tools, which help develop TAP service without coding from scratch. This is very helpful for data publishers. Notably, tools like SkyQuery (Dobos et al., 2012), TAPHandle (Michel et al., 2014), DaCHS (Demleitner, 2014), etc., can be used to perform advanced cross match on various data sources accessed using TAP with scientific cloud storage (e.g. SciDrive Mishin et al., 2014). With the help of DataLink (Dowler et al., 2013), an emerging VO protocol, it is now possible to bind data returned by TAP queries with any kind of data or services. This mechanism will enable new services and enrich a TAP-served data collection by exposing complementary information. For example, an image dataset can be available by TAP and bound to alternate representations, like previews, or enriched with instrumental response maps. For huge data cubes, TAP results could be bound to cutout services that select and extract a subset.

Acknowledgments

We would like to thank all the people in IVOA who defined the TAP standard. We would also like to thank all the developers who worked on implementing this standard to write TAP web services and data providers who are using these to serve their data. Thanks to all those who are involved in writing dedicated clients for a TAP service or accommodating TAP feature support in their existing tools. We would like to thank all the participants who took part in "TAP Questionnaire" survey on the IVOA forum in April 2014. We would also like to thank Robert Hanisch for editorial assistance.

This work was supported by NSF Cooperative Agreement AST-0834235.

Appendix A. Publication tools

See Table A.1.

Appendix B. Generic clients

See Table B.1.

Appendix C. Dedicated clients

See Table C.1.

Glossary

ADQL IVOA Astronomical Data Query Language, It is derived from the Structured Query Language (SQL) language dedicated to support generic and astronomy specific operations. http://www.ivoa.net/documents/latest/ADQL.html.

BNF In computer science, BNF (Backus Normal Form or Backus–Naur Form) is one of the two[1] main notation techniques for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets and communication protocols; the other main technique for writing context-free grammars is the van Wijngaarden form. They are applied wherever exact descriptions of languages are needed, such as in official language specifications, manuals, and textbooks on programming language theory.

DAL Data Access Layer Interface Version 1.0 http://www.ivoa.net/documents/DALI/.

Defines the common features of an IVOA DAL service, the core set of common resources and common parameters, success and error responses, and DAL service registration.

DataLink http://www.ivoa.net/documents/DataLink/index.html .

A mechanism that provides the linking of data discovery metadata to services that provide access to further detailed metadata, the data itself, and services that perform operations on the data. The DataLink web service capability is one such service; it is an intermediate data access service that connects discovered datasets to the downloadable data files, services that can act upon the data files, and links to related resources. to bind datasets either with advanced metadata or with services.

Database A database schema is a container grouping objects such as tables, views or stored procedures etc. The schemas can be used to help logically organize the data or to grant specific user privilege for a subset of the database.

GET HTTP GET is a common method used to interact with a Web service. The GET method requires the parameters to be appended to the service url

Table B.1
Generic clients.

Generic clients	Description
STILTS tapquery http://www.star.bristol.ac.uk/~mbt/stilts/sun256/tapquery.html M Taylor—Bristol University (UK)	Java library: Stilts is a set of command-line tools based on the Starlink Tables Infrastructure Library. It deals with the processing of tabular data.
TOPCAT TAP client http://www.star.bristol.ac.uk/~mbt/topcat/sun253/TapTableLoadDialog.html M Taylor—Bristol University (UK)	JAVA Standalone client: TOPCAT is an interactive graphical viewer and editor for tabular data.
TAPHandle http://saada.unistra.fr/taphandle/ Strasbourg Observatory (Fr)	Universal Web portal for TAP services.
GAVO DC Virtual Observatory Libraries http://soft.g-vo.org/subpkgs GAVO (Ge)	Python library: Some parts of DaCHS useful for other VO software are available as separately installable libraries.
TAP shell http://vo.ari.uni-heidelberg.de/soft/tapsh University of Heidelberg (Ge)	The TAP shell provides a command line interface to querying TAP servers, complete with metadata management and command line completion.
VESPA http://voparis-europlanet-new.obspm.fr/planetary/data/epn/query/all/ VO PARIS (Fr)	Web client for solar and planetary data access.
jc client http://jvo.nao.ac.jp/jcclient NAO (Jp)	JVO command line tool.
VODb https://sites.google.com/site/virtualobservatorydatabase/credits University Observatory Munich (Ge)	VODb is a Java desktop application that simplifies the process of extracting data from astronomical databases.
Seleste http://cda.cfa.harvard.edu/seleste/ CFA(US)	Seleste is a Java tool that uses VO standards and protocols to access archives that expose tabular data and query them using a uniform interface.
Astrotaverna http://amiga.iaa.es/p/290-astrotaverna.htm Instituto de Astrofisíca de Andalucía	Taverna (workflow management system) plugin integrating discovery and access to TAP services.

Here are the client applications and libraries on which developers can build a TAP client application.

Table C.1
Dedicated clients.

Dedicated clients	Description
Vizier TAP interface http://tapvizier.u-strasbg.fr/adql/ CDS (Fr)	A complete web application dedicated to the Vizier TAP service. A table selector help to locate Vizier tables.
Simbad TAP interface http://simbad.u-strasbg.fr/simbad/sim-tap CDS (Fr)	Complete web application dedicated to the SIMBAD TAP service. Query templates are provided.
CADC TAP interface http://www1.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap/ CADC (Ca)	Simple page helping to discover the CADC TAP service with various examples.
GAVO TAP interface http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap/async GAVO (Ge)	Basic html form to submit ADQL jobs on the GAVO TAP node.

TAP dedicated client applications developed in the VO community.

(e.g; http://server/service?param1=value&...). Practically, knowing a GET URL (bookmarked for instance) suffices to run the web service. GET is typically dedicated to services returning data. The GET method requires decoding/encoding operations to serialize the parameters in a way compatible with the URL syntax.

HTTP The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol (source http://www.w3.org/Protocols/).

ObsCore ObsCore refers to the core components of the Observation Data Model Core their implementation in the Table Access Protocol:
The Observation Data Model is a generic data model describing an observation data product that is used to perform data discovery across diverse data centers for observations of interest. http://www.ivoa.net/documents/ObsCore/.

POST HTTP POST is a common method used to interact with a Web service. In the POST method, the parameters are submitted to the service. POST is dedicated to services updating the resource content or to services accepting large or complex parameter sets.

RDBMS A Relational Database Management System is a type of database management that stores data in

the form of tables. Relational databases require few assumptions about how data is related or how it will be extracted. As a result, the same database can be viewed in many different ways.

REST Representational State Transfer is a key design idiom that embraces a stateless client–server architecture in which the web services are viewed as resources and can be identified by their URLs. REST is an analytical description of the existing web architecture (source http://www.oracle.com/technetwork/articles/javase/restful-142517.html).

SIA Simple Image Access, protocol has capabilities for the discovery, description, access, and retrieval of multi-dimensional image datasets, including 2-D images as well as data cubes of three or more dimensions. http://www.ivoa.net/documents/SIA/.

SSA Simple Spectral Access, defines a uniform interface to remotely discover and access one dimensional spectra. http://www.ivoa.net/documents/SSA/.

TAP_SCHEMA All TAP services must support a set of tables in a schema named TAP_SCHEMA that describe the tables and columns included in the service. Details in section 2.6 of TAP standards document. http://www.ivoa.net/documents/TAP/20100327/REC-TAP-1.0.html.

UCD The Unified Content Descriptor (UCD) is a formal vocabulary for astronomical data that is controlled by the IVOA. The vocabulary is restricted in order to avoid proliferation of terms and synonyms, and controlled in order to avoid ambiguities as far as possible. It is intended to be flexible, so that it is understandable to both humans and computers. UCDs describe astronomical quantities with string labels; these are built by combining words from the controlled vocabulary. http://www.ivoa.net/documents/REC/UCD/UCD-20050812.html.

UTYPE A UType is a label used as an identifier for a concept defined within an IVOA data model. Utypes are semantically equivalent to a URI or XPath in XML. (source http://www.ivoa.net/documents/Notes/UTypesUsage/20130213/NOTE-utypes-usage-1.0-20130213.html).

UWS The Universal Worker Service (UWS) pattern defines how to build asynchronous, stateful, job-oriented services. It does so in a way that allows for wide-scale reuse of software and support from software toolkits. http://www.ivoa.net/documents/UWS/20140527/WD-UWS-1.1-20140527.html.

VOSI The Virtual Observatory Standard Interface, VOSI, is the standard that defines the basic functions that all VO services should provide in order to support management of the VO. Details are available at http://www.ivoa.net/documents/VOSI/20110531/REC-VOSI-1.0-20110531.html.

VOTable The VOTable format is an XML standard for the interchange of data represented as a set of tables. In this context, a table is an unordered set of rows, each of a uniform structure, as specified in the table description (the table metadata). Each row in a table is a sequence of table cells, and each of these contains either a primitive data type, or an array of such primitives. VOTable is derived from the Astrores format, itself modeled on the FITS Table format. VOTable was designed to be close to the FITS Binary Table format. http://www.ivoa.net/documents/VOTable/20130920/REC-VOTable-1.3-20130920.html.

WebService A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing. Source http://www.w3schools.com/Webservices/default.asp.

XML XML stands for eXtensible Markup Language. http://www.w3schools.com/xml/.

References

Bonnarel, F., Fernique, P., Bienaymé, O., Egret, D., Genova, F., Louys, M., Ochsenbein, F., Wenger, M., Bartlett, J.G., 2000. The ALADIN interactive sky atlas. A reference tool for identification of astronomical sources. Astronom. Astrophy. Suppl. 143 (April), 33–40. http://dx.doi.org/10.1051/aas:2000331.

Demleitner, M., 2014. The DaCHS multi-protocol VO server. In: Manset, N., Forshay, P. (Eds.), Astronomical Society of the Pacific Conference Series, vol. 485. p. 309.

Dobos, L., Budavári, T., Li, N., Szalay, A.S., Csabai, I., 2012. SkyQuery: an implementation of a parallel probabilistic join engine for cross-identification of multiple astronomical databases. ArXiv e-prints, June.

Tody, Doug, Dolensky, Markus, McDowell, Jonathan, Bonnarel, Francois, Budavari, Tamas, Busko, Ivan, Micol, Alberto, Osuna, Pedro, Salgado, Jesus, Skoda, Petr, Thompson, Randy, Valdes, Franceso, 2012. In: Doug, Tody (Ed.), Simple Spectral Access Protocol Version 1.1, IVOA Recommendation. http://www.ivoa.net/documents/SSA/.

Dowler, P., Bonnarel, F., Michel, L., Donaldson, T., Languignon, D., 2013. Datalink. IVOA Working Draft. URL: http://www.ivoa.net/documents/DataLink/.

Dowler, P., Demleitner, M., Taylor, M., Tody, D., 2014. IVOA Recommendation: DALI: data access layer interface version 1.0. ArXiv e-prints, February.

Dowler, P., Rixon, G., Tody, D., 2011. IVOA Recommendation: table access protocol version 1.0. ArXiv e-prints, October.

Louys, M., Bonnarel, F., Schade, D., Dowler, P., Micol, A., Durand, D., Tody, D., Michel, L., Salgado, J., Chilingarian, I., Rino, B., Santander-Vela, J.D., Skoda, P., 2011. IVOA Recommendation: observation data model core components and its implementation in the table access protocol version 1.0. ArXiv e-prints, November.

Michel, L., Nguyen, H.N., Motch, C., 2006. How to publish local data into the VO with Saada. In: Gabriel, C., Arviset, C., Ponz, D., Enrique, S. (Eds.), Astronomical Data Analysis Software and Systems XV. In: Astronomical Society of the Pacific Conference Series, vol. 351. p. 25.

Michel, L., Louys, M., Bonnarel, F., 2014. Browsing TAP services with TAPHandle and DataLink. In: Manset, N., Forshay, P. (Eds.), Astronomical Society of the Pacific Conference Series, vol. 485. p. 15.

Mishin, D., Medvedev, D., Szalay, A.S., Plante, R.L., 2014. Data publication and sharing using the SciDrive service, in: American Astronomical Society Meeting Abstracts, vol. 223, January, p. 255.26.

Motch, C., Michel, L., Pineau, F.-X., 2009. The XCat-DB: a multi-wavelength view on the 2XMMi catalogue. In: Bohlender, D.A., Durand, D., Dowler, P. (Eds.), Astronomical Data Analysis Software and Systems XVIII. In: Astronomical Society of the Pacific Conference Series, vol. 411. p. 466.

Ochsenbein, F., Williams, R., Davenhall, C., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A., Taylor, M.B., Wicenec, A., 2011. IVOA Recommendation: VOTable format definition version 1.2. ArXiv e-prints, October.

Plante, R., 2007. Chapter 36: The astronomical dataset query language (ADQL). In: Graham, M.J., Fitzpatrick, M.J., McGlynn, T.A. (Eds.), Astronomical Society of the Pacific Conference Series, vol. 382. p. 381.

Taylor, M.B., 2014. Visualizing large datasets in TOPCAT v4. In: Manset, N., Forshay, P. (Eds.), Astronomical Society of the Pacific Conference Series, vol. 485, p. 257.