Dancing the VOltz

Hendrik Heinl, Dave Morris

May 24, 2021

Abstract

The shear amount of data available for astronomy today, offered by distributed services not only demands a specific set of skills from (data) scientists: it's obvious that automated processes and machine readable (machine "learnable") interfaces and standards are necessary to combine data from different sources, instruments and messengers. The IVOA defines such standards and protocols and introduces them to the community.
The process of the development of standards and their evolution is an ongoing task. At two Interoperability meetings (or "Interops") per year, the IVOA members assemble. These meetings are rather a working meeting than a conference and understanding the many terms and abbreviations used throughout the sessions is quite challenging for a newcomer.
This tutorial shall give an introduction to the organisation of the IVOA, the working process, and also the terms used. It shall give an overview on existing standards, those currently in development and make it easier for newcomers to get into the IVOA meeting. It is not meant as an introduction to the VO.
This tutorial is presented at the newcomers session for the IVOA Interoperability meeting in May 2021

Software: TOPCAT V4.8, Aladin V11.0PyVO, PyVO

1 Starting out

We will use the VO tools TOPCAT and Aladin. These client software can be found here:

Aladin: `http://aladin.u-strasbg.fr/`

TOPCAT: `http://www.star.bris.ac.uk/~mbt/topcat/`

We will also use PyVO and Astropy. For Linux users we recommend to use your distribution packages. In debian this would be `apt get install python3-astropy python3-pyvo`. For other OS you can follow the installation instructions:

Astropy: `https://docs.astropy.org/en/stable/install.html`

PyVO: `https://pyvo.readthedocs.io/en/latest/`

A comprehensive collection of the examples in this tutorial as well as installation scripts you will find on github.

2 TOPCAT, SCS and TAP

This tutorial will work along the use case of searching for neutrino data from a VO-service and combine these with data from catalogues using Topcat. In the second part we will be using PyVO to find and access observational dataproducts on different VO services. We are well aware that scientifically speaking this does not make a lot sense (yet), but this tutorial is meant to give an overview over the existing VO standards and protocols, and how these are developed within the IVOA Working Groups and Interest Groups.

▷ **1** *Finding Neutrino data in the VO-registry* –

We start this use case with searching and accessing neutrino data that is available on VO services. Start Topcat and click on VO → Cone Search. In the Cone Search window enter `neutrino` as Keywords and click on Find Services. After a few moments you see the results coming in as a list of services providing data related to the keyword. You see that the list provides you with some information of what the data of the service contains. In the description you also get information about the different data provides.

VO Registry

The Registry is the central point for searches for services in the VO. VO services identify themselves to the Registry, providing metadata about the data they serve and the service protocols they support. Thus, the Registry is the entry point for data discovery in the VO.

Mark different rows and watch the URL change in the field below. This meta data is meant to give you an idea of what you can expect from the services and if they are of interest for you.

With this little example you already get an idea of what the VO standards and protocols are about: standardize publishing, searching and accessing data and making it reusable.

Exercise: Of course you may not be interested in neutrino data. Try to find the messengers that are of importance for you instead: try radio, infrared or gamma ray, or whatever is suitable for you. Note: here the registry is searching for SCS services only. The searches for other service protocols (SIA, TAP, SSAP) work similar.

▷ **2** *Retrieve neutrino data through a Cone Search* –

For our example, we select the service provided by km3net by marking the service by the title ANTARES 2007-2017. We will get a bunch of neutrino events to use in the next steps. In Cone parameters for RA enter `170`, for Dec enter

25 and and give a Radius of 30 degrees. Then click on OK. Within seconds the data will be loaded into Topcat and you see the table appear in the main window of Topcat.

The protocol we used is the Simple Cone Search (SCS).

SCS

SCS (Simple Cone Search) defines a standard web service interface for requesting data based on a cone projected on the sky described by a sky position and an angular distance. The protocol defines the query parameters the service understands and the format of the response. This enables users to query multiple services using the same criteria. Each service returns a machine readable VOTable document listing the astronomical sources or objects which are within the search criteria.

The table we received from the SCS is a VO-Table, which not only contains the actual data, but also some metadata.

VOTABLE

VOTable is the standard data exchange format for tabular data in the VO. The VOTable standard defines an XML schema that combines data types, units, UCDs and data model references to provide machine readable metadata describing each column in the table, enabling data processing and display software to understand how to use it.

So let's have a look at it. We click on Views → Column Info. Here you can get some information about the columns. You see that the data provides us with a modified Julian Date of when the neutrino event was ovserved. If we scroll a bit to the right, we see the UCD (Unified Content Descriptors). In charge of UCDs is the Semantics WG.

We can plot some of the data with Graphics, but so far we wouldn't have much more than positions to see.

The table we received from the SCS is a VO-Table, which provides not only the actual data, but also some metadata.

▷ **3** *Adding catalogue data to the neutrino candidates with TAP* –

In this step we want to combine our neutrino data with catalogue data from SDSS. Catalogue data typically is table data, so we will use the Table Access Protocol and the corresponding query language ADQL to do so. There is a lot more to say about TAP and ADQL than we will do here. A comprehensive ADQL introduction is linked in the References.

UCDs – VO semantics

UCDs (Unified Content Descriptors) describe the content of table columnms in a machine readable way. Thus, the machines "know" something about the data and you can make use of this in your programs. The data type of a column may describe it as a floating point number, the units may say it is measured in degrees, but the UCD allows us to identify it is a *right ascension position coordinate*, enabling data processing and display software to understand how to use it.

In Topcat go to VO → TAP. The TAP window will open and after a few moments you will see a list of services in the window. Analogously to the Cone Search window, you can use the TAP window to search the VO-registry for data. At Keywords we will enter `SDSS` to search for services providing us with those data. Often you will find the most recent catalogue data from dedicated services related by the publishers of the catalogues. You may also find them on the TAP Service of Vizier. Many Services providers try to anticipate which data combination might be of interest for the users. So it's not uncommon to find surveys like 2MASS or SDSS on several services.

TAP

TAP (Table Access Protocol) defines a standard web service interface for querying tabular catalogs. The protocol defines how to represent the query parameters and the available response formats. This enables users to query multiple services using the same client software and query language.

As for SDSS you see a few options to chose from. We will select the GAVO data center (Note: the reasons here is due to the performance of the tutorial: we want query results to come in fast. Real science takes time and it is completely acceptable to run queries for several hours). Click on Use Service to get to the TAP window of the Service.

Give it a few moments to load the meta data of the service into the TAP window. On the left you see a tree of the catalogues and tables in this service, on the right is the metadata browser on these tables. Mark any table and see what the description of the table and the columns tell you about the data in the table. This meta data still is part of the TAP standard and is very helpful for data discovery and of course for the data access using ADQL.

In the upper left of the meta data at Find: enter `sdss` to find the table containing this survey. In this example we just want to collect the identifier of an object, the position in ra and dec and the colours in the u, g, r, i and z band.

We will make use of the TAP feature TAP UPLOAD, to join our local table with results from the remote service. In the lower window at ADQL Text click on Examples → Upload → Upload Join. You see an example ADQL query appearing in the field:

```
SELECT
   TOP 1000
   *
   FROM sdssdr7.sources AS db
   JOIN TAP_UPLOAD.t1 AS tc
   ON 1=CONTAINS(POINT('ICRS', db.ra, db.dec),
                 CIRCLE('ICRS', tc.ra, tc.decl, 5./3600.))
```

Now at the first glance that may look confusing, so let's go through it step by step, customizing it to fit our science case as we go. Each ADQL query starts with `SELECT` followed by specifications of "what" to select, what to do with the selected records and how to return the results. `TOP 1000` means the first 1000 data records will be returned, which match the whole query. With `*` we select all columns of matching data records. Here will will modify the query, because we are not interested in all of the columns of SDSS. Instead of `*` we will write: `antares.*, sdss.objid, sdss.ra, sdss.dec, sdss.u, sdss.g, sdss.r, sdss.i, sdss.z`. This means we want to keep all columns from our uploaded table (which we will call `antares`) and add the columns from the remote table (which we will call `sdss`). The phrase `FROM sdssdr7.sources AS sdss` specifies the table of the TAP service we want to query `sdssdr7.sources` and the short name we want to use for it `sdss`. The next lines are the query conditions. In our example we use the ADQL built in functions that defines the upload join. The first part `JOIN TAP_UPLOAD.t1 AS antares` specifies that we want to join the data from SDSS with data in the table of neutrinos from Antares that we uploaded alonfg with the query, and the short name we want to use for it. Then we describe the criteria for joining the data by defining a circle around each object in our table of Antares neutrinos, and asking the database to check if any sources in the SDSS table lies within any of these circles. A little confusing may be the `1=CONTAINS`. This is due to the boolean result returned by the function `CONTAINS`. A result of `1` means `true` whereas `0` means `false`. `POINT` expresses a point, the right ascension and the declination in degrees. Our query takes the coordinates from the table using the columns `ra` and `dec`. Analogously `CIRCLE` expresses a cone in space, taking `ra` and `decl` from our local table. The last entry defines the radius of the circle in an angle in decimal degrees. The whole query looks like this:

```
SELECT
```

```
TOP 1000
antares.*, sdss.objid, sdss.ra, sdss.dec, sdss.u, sdss.g, sdss
    .r, sdss.i, sdss.z
FROM sdssdr7.sources AS sdss
JOIN TAP_UPLOAD.t1 AS antares
ON 1=CONTAINS(POINT('ICRS', sdss.ra, sdss.dec),
              CIRCLE('ICRS', antares.ra, antares.decl,
                  10./3600.))
```

Click on Run Query and the result should come in rather fast. Again have a look at the new table. You see that we added a few columns, but it seemed that we lost some rows. This is due to the query: the database did only return those data records that matched our conditions. Not matching records are discarded.

ADQL

ADQL (Astronomical Data Query Language) is a query language based on a subset of SQL used to query tabular data in TAP services. The ADQL standard defines a common subset of SQL that can be used to access data in all the common relational database platforms used in astronomy. This enables users to query multiple services using the same query without having to worry about the specific dialect of the relational database platform hosting the data.

Exercise: You can control the ADQL JOIN behaviour by using the alternatives `LEFT OUTER JOIN`, `RIGHT OUTER JOIN` or `FULL OUTER JOIN`. The default command is `INNER JOIN`. Play around with these options, they might become usefull in future.

With the colors of SDSS we could now analyze the neutrino sources and do more reasonable science.

▷ **4** *Getting a Table with DataLink* –

Once you have a certain level of interoperability, you can go on and add more functionality and interactivity between clients, servers or in between clients. One of the standards developed for this is Datalink. Some services and clients can already consume datalinks. Luckily, the GAVO service, Topcat and Aladin are capable of this. So at this point make sure that Topcat and Aladin are running. Then go to the Topcat TAP window, select the GAVO service and run the following query to obtain a datset containing DataLinks:

```
SELECT
   TOP 10000
   *
   FROM sdssdr7.sources AS sdss
   JOIN TAP_UPLOAD.t1 AS antares
   ON 1=CONTAINS(POINT('ICRS', sdss.ra, sdss.dec),
                 CIRCLE('ICRS', antares.RA, antares.Decl,
                     5./3600.))
   JOIN lsw.plates as plates
   ON 1=CONTAINS(POINT('ICRS', plates.centeralpha, plates.
       centerdelta),
                 CIRCLE('ICRS', antares.RA, antares.Decl, 1.))
   AND plates.accsize<200000000
```

Note that we added another `JOIN ON` with a geometry defined by the antares data, but also on the remote table lsw.plates. Here we want to get some information about scanned plates from the "Landessternwarte Heidelberg´´. The snippet `AND plates.accsize<200000000` is added because some of the FITs files exceed 1 GB of data volume, which would take quite some time to download. So for the sake of the tutorial, we limit the filesize. Again, click on Run Query. It will take a few seconds until the data is available. Then go Topcat's main window, click on Views → Activation Actions. A new window will open. On the left side click on Invoke Service and check the box next to it. See if in the main window as Action `View DataLink Table` is selected. Now Topcat is prepared to interprete the DataLink block of the VOTable. Click on the flash-button below to "Invoke Action on first row".

Datalink

The Datalink standard connects meta data of datasets to the data, related data products, and web service capabilities that can act upon the data, without having to download the whole dataset.

▷ **5** *Using DataLink* –

A new window will open and show you the DataLink Table. Here you find four rows: `#auxiliary`, `#proc`, `#this`, and `#preview`. If you scroll through these four entries, you notice, that `#auxiliary`, `#this`, and `#preview` contain links to fits file, and a possible Action is `View image internally`. With this, you can open a small window in Topcat that will either show you a 1/4 scaled image, the whole dataset, or a small cutout of the image. `#proc` is different, though. This one let's you use the SODA standard to make your own cutout of the image. Since we actually don't know much about it, we won't go down this

road. Instead, let's use a proper image analysis software to have a look at the images: Aladin. Mark `#auxiliary` and in the drop down menu at the buttom of the window selec at Type: `FITS_Image` and next to it at Action `Send FITS Image`. Then switch to Aladin and see how the image is loaded. You can open the Topcat Data Display and select a different row of our table, and then repeat the step of sending the FITS-image to Aladin. Note that whenever you hover over the "SAMP" planes in Aladin, how it will compute the covered area of the image and display it in the main window. If your screen is big enough, you can easily display Topcat and Aladin on the same screen and switch between the too.

Exercise: If you are interested in more interoperability, send the table we obtained in the earlier tutorial step to Aladin via SAMP and see where the sources of our neutrinos are in the images. We don't claim that you actually see the neutrino sources in the images. But if you do, let us know, because it would be multi messenger astronomy! Select data points in Aladin and watch how they are highlighted in Topcat.

Exercise: If you feel ready to get into the depths of the VO: save the Table of this last tutorial step to a VO-Table and look at it in your favourite text editor. Can you figure out which bit is defining the DataLink functionality ?

3 PyVO and ObsTAP

PyVO is the Python API to the VO standards. You can use it for data discovery and data access in the same way as you would use VO clients like Topcat or Aladin. But you can also use it embedded in your own code, so that you may access data remotely from an automated script. We will show with three simple (and one not so simple) scripts how you can use PyVO in the multimessenger context.

▷ **6** *Example 1: PyVO and TAP* –

Have a brief look at the few lines in example1.py. Note, that two lines are necessary to supress warnings – services and clients often are a bit off, especially in regard of recent standards. They still work, but warnings will be raised. We don't want to pretend that everything in the VO is perfect, but within a tutorial, warnings might be confusing. If you are curious what's going on, simply comment the according lines and run the script. The script performs a very simple query on the GAVO obscore service. Have a look how the service object is built by `pyvo.dal.TAPService()`. This is the convention within PyVO:

`pyvo.dal.SERVICETYPE(parameters)`. The actual search is performed with the service object method `search`. This will send the query to the server, and the last line saves the resulting VOTABLE into the same folder as the script.

▷ **7** *Example 2: PyVO and SCS* –

This example shows analogously the Simple Cone Search from PyVO. The service object is built following the convention introduced in the last tutorial step, and as you see, we again use the service provided by km3net to obtain a list of neutrino events. Here we chose a much smaller radius though. We want to keep the following steps fast. Note the last lines of the script: we save the result to a local file, but we also use SAMP to broadcast the result to topcat. To make this work, you need to make sure that topcat is running before you run the script. You will find the result as a table in topcat now.

SAMP

SAMP (Simple Application Messaging Protocol) enables astronomy applications on the same desktop or laptop machine to share data with each other. It also enables applications to notify each other about what data points a user has selected, enabling the two applications to coordinate their display and selection views.

▷ **8** *Example 3: PyVO and Obscore* –

Example now is a bit closer to the real world. The script will make use of the neutrino data we get from example 2, which we saved locally to the hard drive. So we take different steps from here: we load the data from the folder exampl2. Note that we have a fallback exception in the code, in case something went wrong during the last tutorial step. We also introduce a longer TAP query. From the first part of the tutorial you are familiar with the TAP-upload already. But here we are comparing our local data with a special table on the TAP service: ivoa.obscore. A service providing this table follows a the special standard Obscore, which is dedicated to observational data. Each obscore-service provides this table with exactly defined columns which enables the users to use the same query repeatedly on each obscore-service. Of course there is no garantuee that one will receive any results, but it's making Searches all across the VO possible, and comparable easy. Here we query the gavo dc. only, and sent the results to Topcat.

▷ **9** *Topcat, SAMP and Aladin* –

Look at the results in Topcat. We will have a list of 50 images. Note: it's a table that does not contain the actual images, but urls to the images. Topcat can not deal with these links. Therefore, start Aladin, then go back to Topcat and use SAMP to sent the data to Aladin. In Aladin you now find this table and can download the fits files to your local machine. Hover over the rows to

ObsCore

ObsCore Is a common data model for describing astronomical observations. It defines the core components of metadata needed to discover what observational data is available from a service. Every service that advertises itself as an ObsTAP service must provide this standard view of metadata lsiting the observational data available from that service.. This means that a user can build an ADQL query based on the ObsCore data table and apply the same query to all the ObsTAP services.

see the coverage of the images in relation to the position on the sky you were searching. Scroll a bit left and click on the buttons in the column "Preview" to download smaller Versions of the images in Aladin.

4 Acknowledgements

The Authors thank ESCAPE for making this contribution to the 2021 May Interop possible.

Dave Morris works as a Research Software Engineer at the Royal Observatory, Edinburgh (ROE)

Hendrik Heinl works as Ingénieur d'études at the Centre de Données astronomiques de Strasbourg (CDS)

5 IVOA Entities

The IVOA is organised in "Groups of interest and competence", with some groups being in charge of the development and evolution of standards (mainly the Working Groups), whereas other groups are focus on the practices of data interoperability and the demands of specific groups within the astronomical community (e.g. the Radio astronomy IG or the Solar System IG). IGs help to identify demands, WGs work out if and how VO standards can reply to these demands.

Working Groups

The following WGs are established in the IVOA and are assigned a specific focus on the VO standards. Often a standard will touch the focus of several WGs. Then a joined session of the WGs will be held at the Interop meetings.

Application WG

As the title Applications WG 1 implies, the focus of this WG is on the perspective on the VO from the view of an application developer. Thus, the Sessions include discussions and exchange of VO standards implementation, but also new demands to these standards deriving from the use of applications. The application WG could be linked to any standard or protocol an application might speak or consume, so the involvement is everywhere in the VO. You will find further information here:
https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaApplications

Data Access Layer WG

The Data Access Layer WG is tasked with the definition and formalizatio of standards for remote data access. As such, the WG is in charge of standards like SCS, SIAP, TAP/ADQL and others. A more detailed description can be found here:
https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaDAL

Data Model WG

Data Model Data Model https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaDataModel

Grid and Web Services WG

GWS Grid and Web Services https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaGridAndWebServices

Resource Registry WG

Resource Registry WG Resource Registry WG https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaResReg

Semantics WG

Semantics WG Semantics WG https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaSemantics

Interest Groups

IGs address a specific topic and often are linked to external communities. The Data Curation and Preservation IG has bounds to the Research Data Alliance (RDA), whereas IGs like Radio astronomy IG or Solar System IG link to specific fields in astronomy. IGs connect the IVOA with these different communities and enable outreach, but also input from these communities.

Data Curation and Preservation IG

DCP https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaDCP

Education IG

Edu https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaEducation

Knowledge Discovery IG

KDD https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaKDD

Operations IG

Operations https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaOps

Radio astronomy IG

Radio astronomy IG Radio astro IG https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaRadio

Solar System IG

Solar Systems https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaSS

Theory IG

Solar Systems https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaTheory

Time Domain IG

Time Domain IG Time Domain IG https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaVOEvent

Other Groups and Committees

Eventually there are formal and organisational groups where decisions are prepared and formalized.

Exec

IVOA Executive Exec https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaRepMin

TCG

IVOA Technical Coordination Group TCG https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaTCG

CSP

IVOA Standing Committee on Science Priorities CSP https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaSciencePriorities

TCG

Standing Committee on Standards and Processes SCSP https://wiki.ivoa.net/twiki/bin/view/IVOA/IvoaSciencePriorities

6 Standards

Datalink

ADQL

Obscore

ObsLocTAP

SCS

SAMP

UCD

VOTable

7 References

Demleitner, M. ADQL-Course

Demleitner, M., Becker, S. PyVO Documentation

Fernique, P. Aladin Documentation

Taylor, M. TOPCAT Documentation