



WFAU quarterly - November 2017

Firethorn & Virtual Observatory services





Virtual Observatory

- Contributions to the Time domain interest group
 - Time-series representation, combining work from Jiří Nádvozník and IVOA DM group
- Contributions to the evolution of ADQL
 - ADQL draft 2.1 published on IVOA wiki
 - Compatible with TAP and DALI data types
- Attended IVOA conferences in Shanghai and Chile
 - Presented work on ADQL and VOEvent
 - Contributed to BoF sessions on cloud infrastructure
- Ensuring that IVOA meets LSST needs
 - Working with LSST to promote the changes they need
 - Working to improve consistency between standards





GAIA

- GENIUS demo – Completed
 - Uses Firethorn & DQP to run queries combining Local ROE catalogs and GAIA DR1 at ESAC
- TAP Autocomplete JS library
 - Completed, and being used at the ESA GAIA GUI
- Paper on use of Docker in Astronomy
 - Astronomy and Computing (July 2017, vol 20)
 - Poster at ADASS 2017





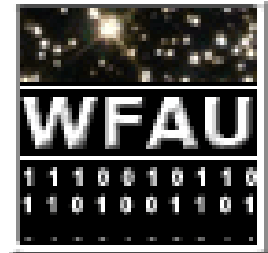
LSST:UK

- Investigating infrastructure to consume and broker alerts to UK users
 - LSST / ZTF data streams (firehose)
- Pyrothorn (Python library used to test Firethorn) being used to benchmark Qserv (LSST database distributed back-end)





Virtual Observatory WFAU services

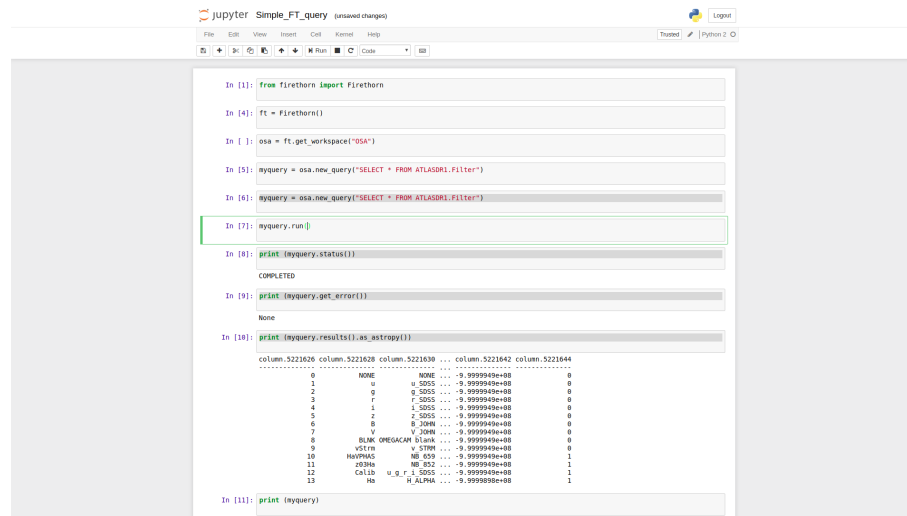


- TAP service running for OSA (ATLAS & ATLAS related databases)
 - Uses Firethorn
- Build and register TAP services for each remaining WFAU archive.
 - WSA
 - VSA
 - SSA
- Replacing old DSA TAP services
- Future work
 - Cone Search service, WFAU publishing Registry, Geometry for our TAP services



Virtual Observatory Firethorn.py

- Building Firethorn python client to access WFAU services via python
- Will be used via Jupyter Notebooks / Jupyter Hub
- Used as an experiment for LSST & useful for WFAU users as another way to access our data



```

In [1]: from firethorn import Firethorn

In [4]: ft = Firethorn()

In [ ]: osa = ft.get_workspace("OSA")

In [5]: myquery = osa.new_query("SELECT * FROM ATLASDR1.Filter")

In [6]: myquery = osa.new_query("SELECT * FROM ATLASDR1.Filter")

In [7]: myquery.run()

In [8]: print (myquery.status())

COMPLETED

In [9]: print (myquery.get_error())

None

In [10]: print (myquery.results().as_astropy())

column.5221626 column.5221628 column.5221630 ... column.5221642 column.5221644
.....
0 NONE ... 0.9999999e+08 0
1 g 0.5055 ... 0.9999999e+08 0
2 g 0.5055 ... 0.9999999e+08 0
3 r 0.5055 ... 0.9999999e+08 0
4 r 1.5055 ... 0.9999999e+08 0
5 z 0.5055 ... 0.9999999e+08 0
6 b 0.3048 ... 0.9999999e+08 0
7 v 0.3048 ... 0.9999999e+08 0
8 BLANK OPERACAN DLink ... 0.9999999e+08 0
9 VSTIN v STIR ... 0.9999999e+08 0
10 HAYMAS MB 059 ... 0.9999999e+08 1
11 z304a MB 052 ... 0.9999999e+08 1
12 ColId u_g_r_i_5055 ... 0.9999999e+08 1
13 Na R_ALMA ... 0.9999999e+08 1

In [11]: print (myquery)

```