Datalink in PyVO

Markus Demleitner[1], *Stefan Becker*[1,2]
[1] *Universität Heidelberg, Astronomisches Rechen-Institut, Mönchhofstraße 12-14, Germany*
[2] SFB 881 "The Milky Way System"
msdemlei@ari.uni-heidelberg.de

Abstract

The Virtual Observatory standard Datalink facilitates the efficient dissemination of complex datasets consisting of multiple artifacts (e.g., science data, calibration files, observation logs, associate plots; see [3]). It also supports the association of server-side preprocessing services (e.g., cutouts, rebinning). Such machine-readable declarations can save large amounts of unneccessary traffic in particular when doing automated processing of multiple, diverse datasets. They also mesh nicely with the focus of the astropy-affiliated package pvVO. We have therefore added support for datalink services and documents to pyVO. This contribution will briefly discuss the compact API we have designed as the glue between user code and datalink.

The Datalink Protocol

The Datalink VO standard [3, 4] provides a file format to group the various parts of a science data product and to declare relationships between them. A datalink document could thus say:

- K9O201.fits is the main observation
- K9O201.jpg is a preview of the observation
- proc/K9O201_masks.fits is a bright star mask
- proc/K9O201_objs.vot is a extracted sources
- proc/K9O201_cmd.png is a CMD derived from the observation

Note that this part of datalink is also very useful as an index file within product packages (e.g., a .tar.gz).

Datalink further defines an access protocol for such documents, and it allows the declaration of server-side processing services. An auxiliary standard called SODA [2] defines some standard operations, in particular for cutouts.

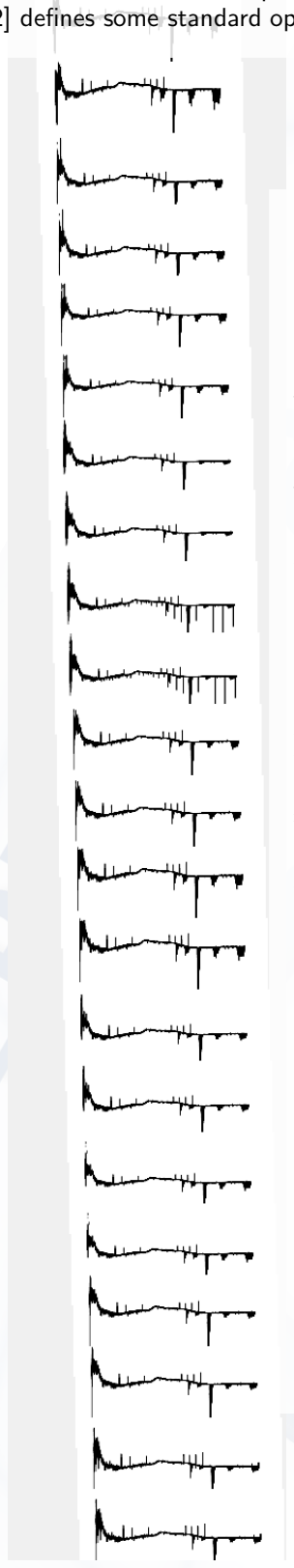

Fig. 1: Spectra for EI Eri, pulled as previews through datalink and arranged in 10 seconds, using example 1.

Datalink Service API

(This is work in progress and subject to change)

Datalink-compliant processing services can be accessed through the `services` attribute of a datalink result. Each element in that exposes the parameters accepted by the service in a `parameters` attribute (which has attributes like `upper_limit`, `lower_limit`, `legal_values`, `ucd`, `unit`, `description`, and, in particular, `value`). From these, arguments to the query method can be built with automatic, astropy-type unit conversion.

For SODA-defined services, there is a shortcut that supports the standards SODA parameters. Its use is illustrated in the following little program that arranges cutouts from historical plates around Mira, which, based on an obscore query, yields Figure 2 with very moderate resource consumption:

```
import math, requests, io, Image
import pyvo
from astropy.coordinates import SkyCoord
from astropy.io import fits

svc = pyvo.dal.tap.TAPService("http://dc.g-vo.org/tap")
roi = SkyCoord.from_name('Mira')

cutouts = []
for rec in svc.run_sync(
    "SELECT access_url, access_format FROM ivoa.obscore"
    " WHERE obs_collection='HDAP'"
    "AND 1=CONTAINS(CIRCLE('ICRS', {}, {}, 0.05),"
    "s_region)".format(roi.ra.deg, roi.dec.deg)):

    processed = rec.processed(
        circle="{} {} {}".format(roi.ra.deg, roi.dec.deg, 0.05))

    pixels = fits.open(io.BytesIO(processed.read()))[0].data
    cutouts.append(
        Image.fromarray(((pixels/float(pixels.max()))*255).astype('uint8')))

per_line = int(math.ceil(math.sqrt(len(cutouts))))
dest_size, stamp_size = 1600, 1600/per_line
montage = Image.new("L", (dest_size, dest_size))
for index, img in enumerate(cutouts):
    montage.paste(
        img.resize((stamp_size, stamp_size)),
        (index/per_line*stamp_size, index%per_line*stamp_size))
montage.save("cutouts.jpg")
```

Example 2

Getting pyVO with Datalink

The Datalink file API is already released with pyVO 0.6.1 (`pip install pyvo` is enough). The service API is currently available by cloning the master git repository [1] and checking out the soda branch.

We expect to provide a release containing the service API in late 2017.

Datalink Files API

The pyVO package has beta-level support for datalink in its development releases.

With this, DAL results have an additional method `iter_datalinks`, which on services supporting datalink, iterates over datalink documents for the results returned.

Datalink documents are modeled as `DatalinkResult` instances. You can iterate over these to obtain all links including their descriptions, media types, etc, or you can directly access specific items through the `bysemantics` method.

Here's some sample code to pull spectral previews for EI Eri and paste them together; this code produces Fig. 1 in about 10 seconds:

```
import io, requests, pyvo, Image
from astropy.coordinates import SkyCoord

svc = pyvo.ssa.SSAService("http://dc.g-vo.org/feros/q/ssa/ssap.xml?")
matches = svc.search(SkyCoord.from_name("EI Eri"), 0.001)
xoff, yfact = 2, 0.7

previews = []
for dl in matches.iter_datalinks():
    prev_url = dl.bysemantics("#preview").next()["access_url"]
    im = Image.open(io.BytesIO(requests.get(prev_url).content))
    previews.append(im)

xsz, ysz = previews[0].size
montage = Image.new("L",
    (xsz+(len(previews))*xoff, int(ysz*yfact*(len(previews)))),
    color=240)
for index, preview in enumerate(previews):
    montage.paste(preview, (index*xoff, int(ysz*yfact*index)))
montage.save("montage.png")
```

Example 1

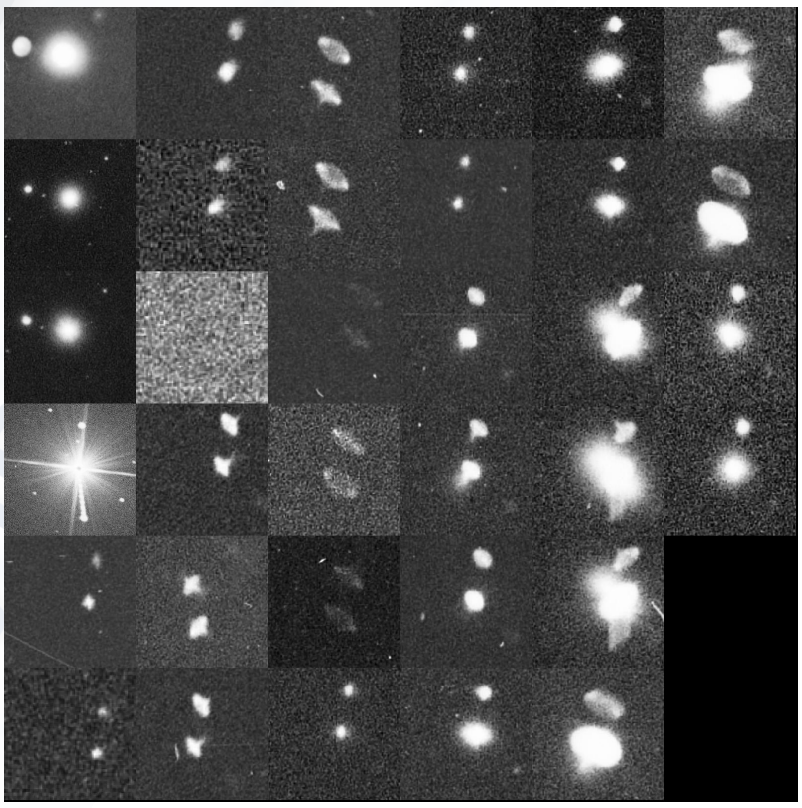

Fig. 2: Cutouts around Mira, pulled from plate scans roughly 1 Gigabyte apiece in a couple of seconds by employing SODA cutouts (example 2).

[1] Stefan Becker et al. pyVO current source code. git repository, 2017.

[2] François Bonnarel, Markus Demleitner, Patrick Dowler, Douglas Tody, and James Dempsey. IVOA server-side operations for data access version 1.0. IVOA Recommendation, May 2017.

[3] Markus Demleitner. Introducing datalink. Poster presented at the ADASS XXV, Sydney, October 2015, October 2015.

[4] P. Dowler, F. Bonnarel, L. Michel, and M. Demleitner. IVOA DataLink Version 1.0. IVOA Recommendation 17 June 2015, June 2015.

Acknowledgement: This work was supported by Sonderforschungsbereich SFB 881 "The Milky Way System" (subproject INF) of the German Research Foundation (DFG).