NASA Astronomical Virtual Observatories

HEASARC • IRSA
NED • MAST

NAVO

NAVO ideas for offering multiple product retrieval options using DataLink

Tess Jaffe, Anastasia Laity, et al.
(NAVOnians and other collaborators)

Zero’th order requirements:

- Do not break anything.
  - It’s invisible if you don’t want to pay attention to it, and existing systems don’t need to change.
- Follow best-practices described for ObsTAP and DataLink implementations.
  - ObsTAP `access_url` columns should be DataLink service calls that allow you to link other products the user might be interested in to the selected row, or do other magic behind the scenes.
  - Two dimensions of information about a single row in an ObsCore table can be specified in the corresponding DataLink result table already: the `semantics` that distinguish the data product from things related to the data product; the `content_type` that distinguish products that are the same data product in a different format.
- Don’t add extra columns unless necessary.
  - But it might be necessary.
- Note: there’s a complex history✳ of how to make the same dataset available through different capabilities, or how to define mirror services. We don’t claim to have understood all of this. And this use case isn’t quite the same.

✳ https://www.ivoa.net/documents/caproles/20190315/NOTE-caproles-1.0-20190315.html

Requirements from a new workflow:

The basic need is to make this workflow possible:

1. Client queries service and gets back an ObsTAP or SxA or DAP result table of matching products.
2. As recommended, the `access_url` is filled with a call to a datalink service (with parameters set by the service for each row).

   There are a couple of possibilities for what happens next, but the functional result is:

3. The client can see that there's a default way to get the data and other options that can be requested or ignored.
4. The user/client chooses one of the offered retrieval locations, either by default or by intent, and sends the corresponding request.
5. The client gets back the data via the selected option.

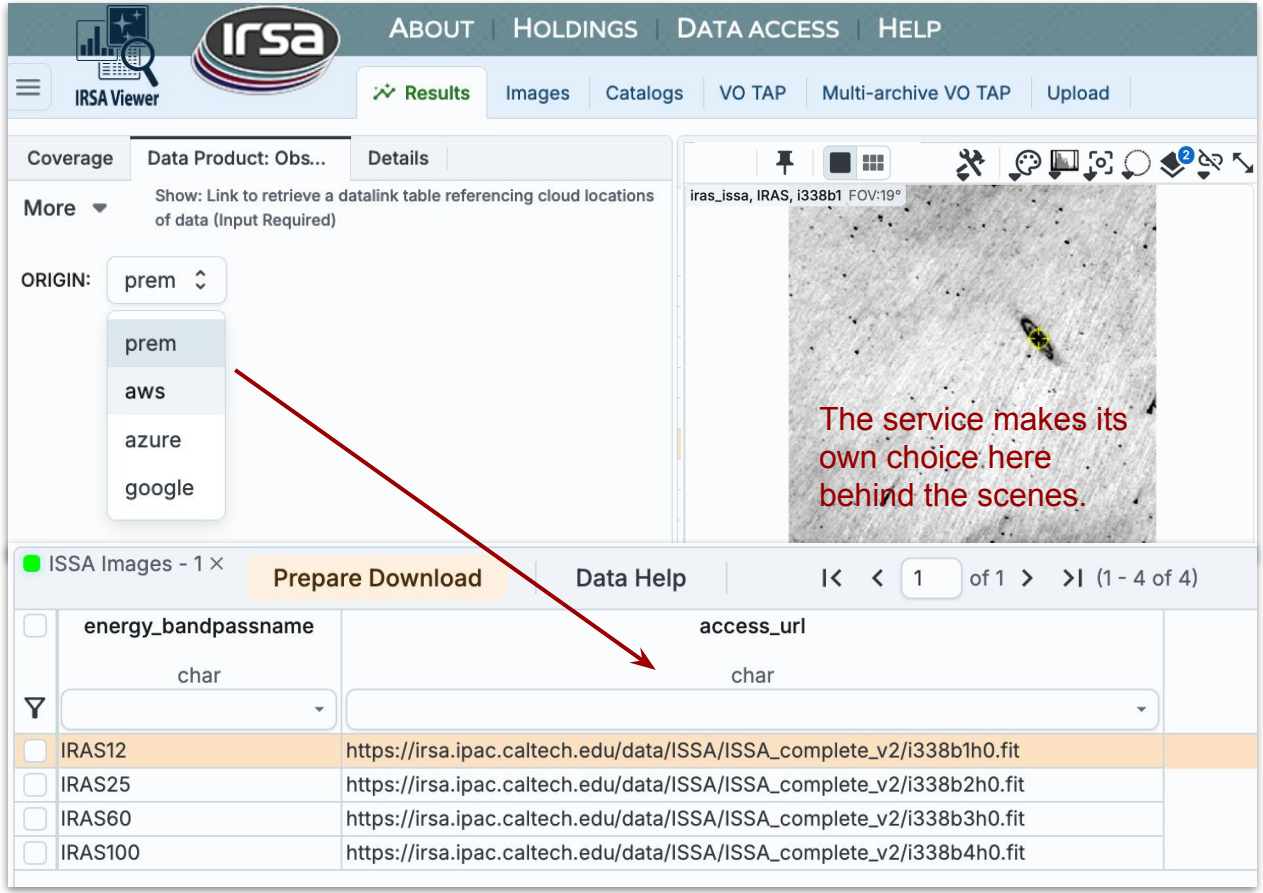
Why do we need this? We may be hosting data in different places, and the access may be more efficient via a non-default option. The server cannot make this decision for the client, because the server cannot know the client's entire context. So the client needs to be able to look at the options and to select one.

Client example in Python (mocked)

```
> import pyvo, astropy
> mysia = pyvo.dal.SIAService('https://example.org/vo/SiaV2?')
> result = mysia.search(query_url, pos=pos, size=0.0)

# Look at options
> print(pyvo.get_data_options(result[0]))
      "prem":  On premises server [default]
      "aws" :  AWS S3 object store in us_east_1
      "gcp" :  Google object store

# get on-prem data by default
> default_handle = pyvo.get_data_product(result[0])
> default_handle.download()

# Get data from AWS option
> aws_handle = pyvo.get_data_product(result[0],origin='aws')
> hdus = astropy.io.fits.open(aws_handle)
      ...
```

- The coder then decides what to do with the pointer given, whether to download all of it or select a subset of the bytes or whatever.
- This is just a simple illustration. Any of this can be coded differently depending on chosen solution. And PyVO may or may not be the place for it, maybe astroquery.

Client example view in Firefly (mockup)

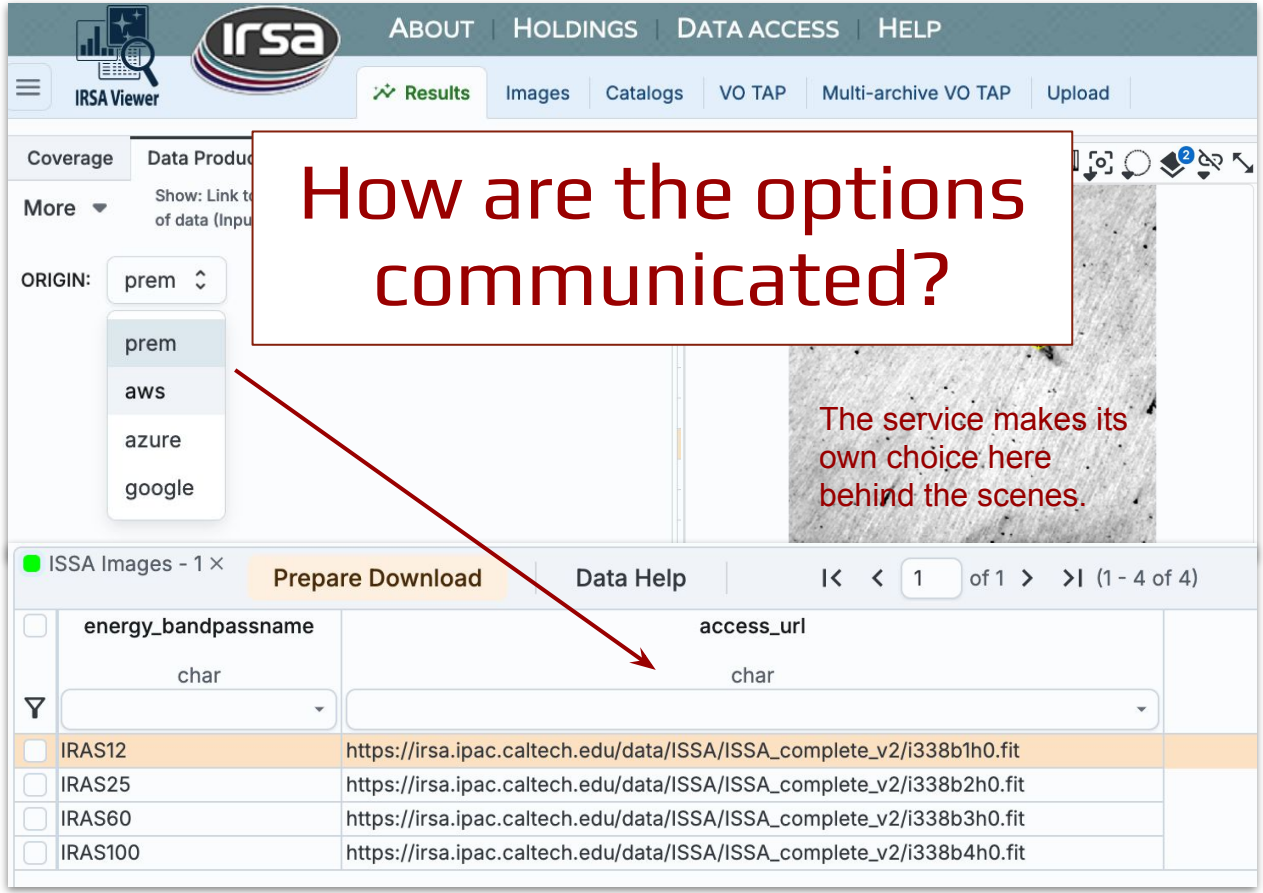


- User searches and finds interesting products.
- User selects one or more of them.
- The client shows the options for retrieving them.
- The user selects an option or takes the default.
- The client gets the address of the product from the selected origin.

★ Note 1: this only makes sense if the user can use the GUI to discover data and then take a list of data addresses elsewhere.

★ Note 2: the user's selection has nothing to do with what the Firefly visualization window on the right chooses to do. The service knows best which option *it* wants.

Client example view in Firefly (mockup)



- User searches and finds interesting products.
- User selects one or more of them.
- The client shows the options for retrieving them.
- The user selects an option or takes the default.
- The client gets the address of the product from the selected origin.

★ Note 1: this only makes sense if the user can use the GUI to discover data and then take a list of data addresses elsewhere.

★ Note 2: the user's selection has nothing to do with what the Firefly visualization window on the right chooses to do. The service knows best which option *it* wants.

Contents of existing metadata columns?

(DataLink result table)

This allows a client to return multiple rows for the identical dataset. The client makes a choice based on the metadata in the columns such as:

- `description`: meant to be human-readable, not a great option to ask a client to read it.
- `semantics`: "#this-aws"? For describing related objects, not meant for *how* to get objects, so maybe not?
- `local_semantics`: same
- `content_type`: the client expects to get the same thing back from all options, so no.
- `content_qualifier`: same?

Semantics example would look like this:

- The ObsTAP result table has a datalink in the `access_url`.

obs_publisher_did	bs_collectio	dataproduct_type	access_url	ac
char	char	char	char	
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	spectrum	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application
ivo://cadc.nrc.ca/mirror/JWST?jw0273:	JWST	image	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application
ivo://cadc.nrc.ca/mirror/JWST?jw0273:	JWST	image	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application

- Firefly requests that and gets a DataLink table with four different options for the identical file.
- They are distinguished by their `semantics`.

Datalink VO Table

ID		access_url	semantics
char		char	char
Same ID		Different URLs	Distinguished by
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	https://cadc-space.nrc-cnrc.gc.ca/fictional/key/to/mission/collection/level2/foobar.fits	#this
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	https://cadc-space.s3.amazon.aws/fictional/key/to/mission/collection/level2/foobar.fits	#this-aws
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	https://cadc-space.azure.ms.com/fictional/key/to/mission/collection/level2/foobar.fits	#this-azure
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	https://cadc-space.cloud.google.com/fictional/key/to/mission/collection/level2/foobar.fits	#this-google

Alternative: service descriptors?

- The ObsTAP result table has a DataLink in the `access_url`. This is a default option. Not shown is that a client could also be giving a service descriptor in the ObsTAP result that tells it retrieving any of these products have options.

obs_publisher_did	bs_collectio	dataproduct_type	access_url	ac
char	char	char	char	
ivo://cadc.nrc.ca/mirror/JWST?jw01199	JWST	spectrum	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application
ivo://cadc.nrc.ca/mirror/JWST?jw0273:	JWST	image	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application
ivo://cadc.nrc.ca/mirror/JWST?jw0273:	JWST	image	https://ws.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/caom2ops/datalink?ID=ivo%3A%2F%2Fcadc.nrc.ca%2Fmi	application

Default

```
<RESOURCE type="meta" utype="adhoc:service" name="
  <DESCRIPTION>
    This datalink service gives access to the raw
    discovered datasets as well as to catalogues o
  </DESCRIPTION>
  <PARAM name="standardID" datatype="char" arraysiz
         value="ivo://ivoa.net/std/DataLink#links-1
```

Other options

- The user selects and gets a DataLink result table with only one `#this` and the `access_url` appropriate to the option they selected:

Datalink VO Table | Prepare Download | 1 of 1 | (1 - 1 of 1)

ID	access_url	semantics
char	char	char
ivo://cadc.nrc.ca/mirror/JWST?jw01199 JWST	https://cadc-space.nrc-cnrc.gc.ca/fictional/key/to/mission/collection/level2/foobar.fits	#this

One URL matching the option requested

DataLink service descriptor options

DataLink service descriptors

This part of the standard is designed to enable this kind of thing.

- Example: minimal spec of four options:

```
<PARAM name="ORIGIN" datatype="char" arraysize="*" value="prem">
    <VALUES>
      <OPTION name="Archive on premises data repository"  value="prem" />
      <OPTION name="Archive AWS Cloud data repository"    value="aws" />
      <OPTION name="Archive Azure Cloud data repository"  value="azure" />
      <OPTION name="Archive Google Cloud data repository" value="google" />
    </VALUES>
 </PARAM>
```

Where is the right place for the service descriptor?

1. In the result table from the original ObsTAP or SxA call.
2. In the result table from calling the `access_url` (for one or more rows), which is a DataLink table.
3. In both.

Considerations:

- Pro for #1: a service descriptor can tell the client how to request all of them in a batch request.
- Con for #1: an result might have data in different rows served via different DataLink services, so attaching a Service Descriptor that only has one base URL is problematic.
    - Possible work-around: DataLink result table has `error_message` telling the client to try one of the other options for this row?
- Pro for #2: The client fetch each row as usual. This calls a DataLink service, which returns a table with a default `access_url` associated with `#this`. Other options are offered in the Service Descriptor that the client can offer or choose to ignore.
- Con for #2: If there is no cloud-options service descriptor in the ObsTAP result, the client/user has to get a datalink result table for each row, look at the service descriptor there, and then send another datalink request to get back the URLs they want.

Back to adding a column?

But this time to the DataLink result table

- Added columns are discouraged unless necessary.
  - Note that an additional columns was defined, `link_auth`, to communicate to the client whether the link requires authentication for similar reasons.
- Previous concept (and current prototypes) added this to the SIA result table because not all archives had implemented DataLink yet. Now we can move it to the DataLink result.
- Reminder: we had proposed one additional column, `cloud_access` whose contents might be:

```
{
    "aws":
        {
        "region": "us-east-1",
        "policy": "free",
        "bucket": "nasa-heasarc",
        "key": "chandra/foo/bar.jpg",
        "access_url": "https://nasa-heasarc.s3.amazonaws.com/chandra/foo/bar.jpg
        }
    "google": {...}
}
```

(TBD which fields would be useful, and the URL could obfuscate the location if needed for paid data.)

- This is just illustrative (in fact it's not exactly what we did), details TBC.

Not mentioned above

- Register separate services from the start? Why we don't much like this option:
  - The discovery workflow doesn't necessarily start with knowing where you want to get the data from.
  - The above questions remain requiring us to agree on metadata to describe the options. The questions just go into a Registry design discussion instead.
  - We already have a similar issue in the Registry when you can get the same dataset from TAP or Cone, or when a service has a mirror. It has added complexity to make the necessary links between the different services.
    - We've already done this painful work, so just use it again?
    - This was painful, let's not do it again?
- Client sends location information with initial request and server decides what URL to give?
  - Users always want to know the options and make their own decision.

Additional questions

- Even without these additional options, is there a recommendation for one or more Service Descriptors with ObsTAP results tables so the client doesn't have to call each `access_url` separately but can send a batch request?
    - Last time I was told modern servers are fast, it isn't a problem sending extra datalink requests. But a bunch of them at the same time?
- If you ask for the AWS datalinks for all of the rows in an ObsTAP result say, and some are on AWS and some are not, then what? I assume the resulting table would have to make a row for that ID with an `error_message` filled in. Do we need a standard … something here?
    - Possibly an argument for adding the data descriptor options to both the original ObsTAP/SxA result and also the datalink result table, so that a client could then try an alternative URL for those rows.
    - But the client could also go back to the original result table (e.g. ObsTAP).
- Other?

Bottom line

- We want to give retrieval *options*.
- This requires a vocabulary *somewhere* to express the options available and how to select among them.
- DataLink is very flexible, and there are several ways we could do this with many combinations of metadata columns and service descriptors.
- We are implementing *something* in the next few months, and the easiest would be to move our extra column to the DataLink result table.
- ????