



Rubin Science Platform: cloud-first, VO-first

Frossie Economou

Russ Allbery

Gregory Dubois-Felsmann



U.S. DEPARTMENT OF
ENERGY



Rubin Science Platform

Portal

Discover data in the browser



[Learn more about the portal.](#)

Notebooks

Process and analyze LSST data with Jupyter notebooks in the cloud



[Learn more about notebooks.](#)

APIs

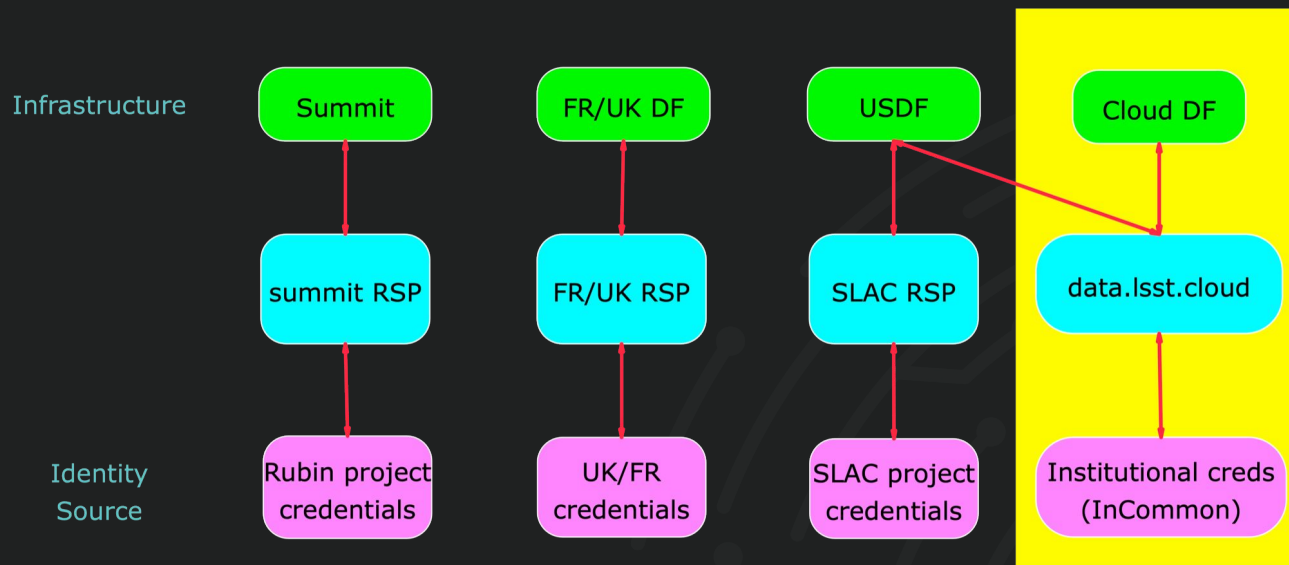
Learn how to programatically access data with Virtual Observatory interfaces



RSP On-prem & On-cloud k8s deployments

- Flagship deployment is and will now remain on Google Cloud
- Being sized for ~10K users
- Cloud choice made for:
 - Scalability
 - Elasticity
 - Isolation
 - Reliability
 - Dev velocity
 - Bringing together separate funding streams
 - Value for money

RSP instances now on 8 distinct sites/infrastructures



Data services available on data.lsst.cloud

Current:

- TAP/ObsTAP (CADC 🙏)
- Image services (implements SODA)
- HiPS (more later)
- DataLink (abstracts image access from ObsTAP results)
- Microservices bouncing TAP results into additional queries via DataLink descriptors (more later)

Planned:

- Registry (definitely)
- VOspace (maybe/eventually)
- ObsLocTAP (probably)

Not Currently Planned:

- SSO Auth v1 (no bearer tokens)
- SSO Auth v2 (unsure as to benefits to our users, design choices unclear to us, need to understand better)

** Rubin implementations in Python*

HiPS service (and GCS)

Constraints:

- Required to restrict access to data rights holders (at least for full resolution)
- Rubin primarily uses object storage
- Data is therefore in a non-public GCS bucket and cannot be served by a normal static file web server

Approach:

- Small Python service serving HiPS-tile files out of GCS (currently by retrieving the whole file and then returning it)
- Alternative was signed URLs, but were worried about problems with redirects
- HiPS list generated by a separate service

Future:

- Serve directly from GCS using Cloud CDN, Cloud Run, and an auth helper

Design goals:

- Only image cutouts as the initial target
- Separate all image manipulation from the mechanics of the service so that it can be independently developed and tested by Rubin pipeline experts
- Build on a general framework that can be reused for other VO services
- GCS used for all storage

Implementation:

- Python FastAPI frontend using dramatiq with a Redis queue and sqlalchemy
- Cutouts performed in a separate worker container based on the Rubin stack
- Slightly complex queue design to separate UWS database operations from backend worker for strict separation of concerns
- Results stored in a temporary GCS bucket and returned with signed URLs

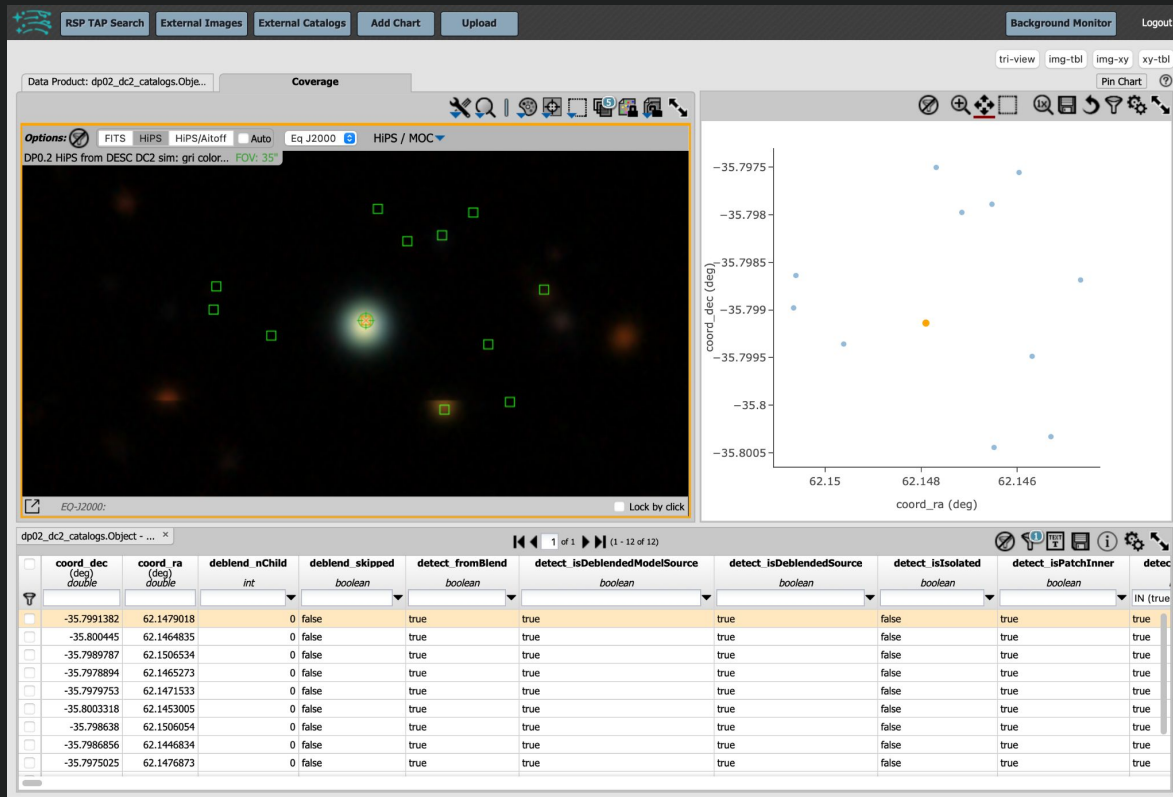
How the portal leverages the DL microservices

As seen from the Portal side:

- Uses VO-standard protocols to explore and visualize data
- Recognizes DataLink service descriptors in TAP results and adds a UI for that service to the result display
- Example: Retrieve a light curve for an object identified by a row in the results of a TAP query
- Recognizes and generates a UI for standard parameter types
 - Circle parameters for a cutout
 - Enumerated parameters such as filter band for a light curve

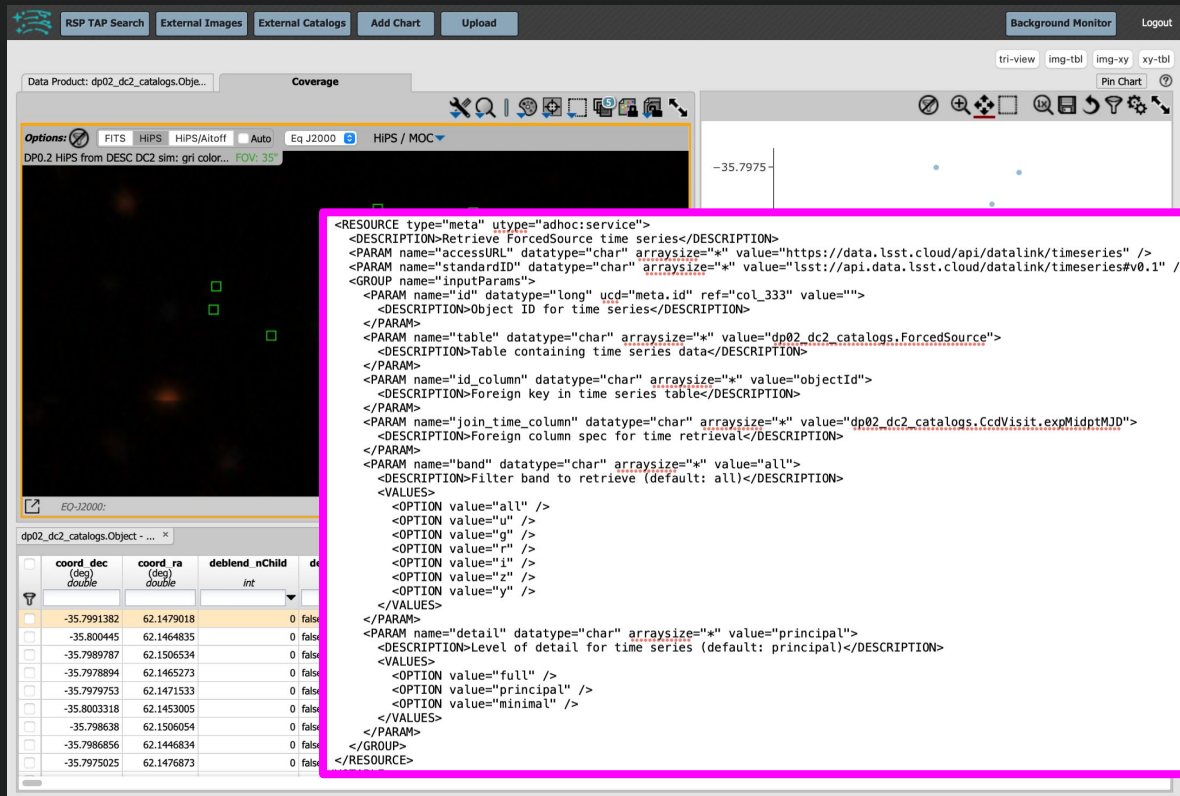
As seen from the API services:

- TAP service is configured to add DataLink service descriptors as needed to results containing designated columns
 - E.g., if Object.objectId is present, add a lightcurve-retrieval link
- Service descriptor points to a route in the datalinker service
- Parameters may be fixed in the URL or specified in the service descriptor if the user should be prompted for them
- Route in the datalinker service (in the current microservices) constructs and redirects to a TAP sync query



1) Perform a search in the Rubin “Object” catalog

(the target is a simulated RR Lyrae star in our Data Preview 0.2 system)



The screenshot shows the DataLink interface with a search for 'dp02_dc2_catalogs.Object...'. The results table is as follows:

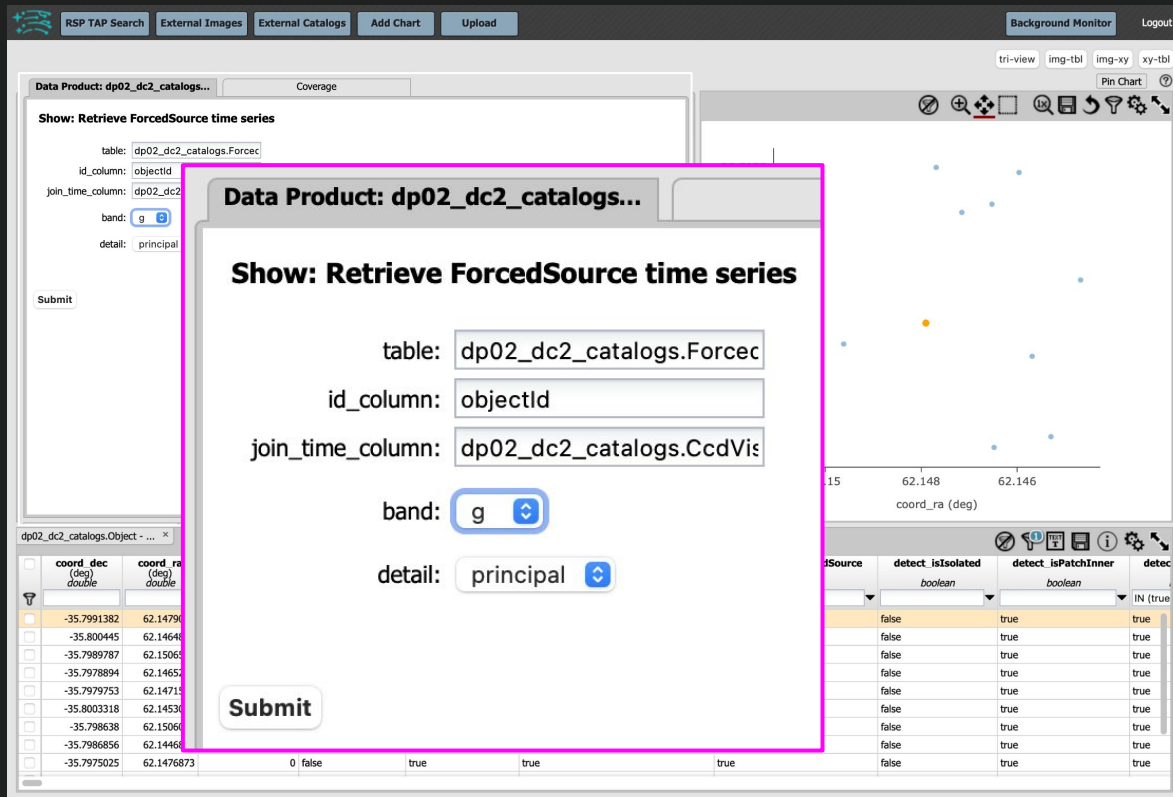
coord dec (deg) double	coord ra (deg) double	deblend nChild int	...
-35.7991382	62.1479018	0	false
-35.800445	62.1464835	0	false
-35.7989787	62.1506534	0	false
-35.798894	62.1465273	0	false
-35.7979753	62.1471533	0	false
-35.8003318	62.1453005	0	false
-35.798638	62.1506054	0	false
-35.7986856	62.1446834	0	false
-35.7975025	62.1476873	0	false

The highlighted XML response is:

```
<RESOURCE type="meta" utype="adhoc:service">
  <DESCRIPTION>Retrieve ForcedSource time series</DESCRIPTION>
  <PARAM name="accessURL" datatype="char" arraysize="*" value="https://data.lsst.cloud/api/datalink/timeseries" />
  <PARAM name="standardID" datatype="char" arraysize="*" value="lsst://api.data.lsst.cloud/datalink/timeseries#v0.1" />
  <GROUP name="inputParams">
    <PARAM name="id" datatype="long" ucd="meta.id" ref="col_333" value="">
      <DESCRIPTION>Object ID for time series</DESCRIPTION>
    </PARAM>
    <PARAM name="table" datatype="char" arraysize="*" value="dp02_dc2_catalogs.ForcedSource">
      <DESCRIPTION>Table containing time series data</DESCRIPTION>
    </PARAM>
    <PARAM name="id_column" datatype="char" arraysize="*" value="objectId">
      <DESCRIPTION>Foreign key in time series table</DESCRIPTION>
    </PARAM>
    <PARAM name="join_time_column" datatype="char" arraysize="*" value="dp02_dc2_catalogs.CcdVisit.expMidptMJD">
      <DESCRIPTION>Foreign column spec for time retrieval</DESCRIPTION>
    </PARAM>
    <PARAM name="band" datatype="char" arraysize="*" value="all">
      <DESCRIPTION>Filter band to retrieve (default: all)</DESCRIPTION>
      <VALUES>
        <OPTION value="all" />
        <OPTION value="u" />
        <OPTION value="g" />
        <OPTION value="r" />
        <OPTION value="i" />
        <OPTION value="z" />
        <OPTION value="y" />
      </VALUES>
    </PARAM>
    <PARAM name="detail" datatype="char" arraysize="*" value="principal">
      <DESCRIPTION>Level of detail for time series (default: principal)</DESCRIPTION>
      <VALUES>
        <OPTION value="full" />
        <OPTION value="principal" />
        <OPTION value="minimal" />
      </VALUES>
    </PARAM>
  </GROUP>
</RESOURCE>
```

1) Perform a search in the Rubin “Object” catalog

→ TAP service returns result annotated with **service descriptor** for light-curve query based on objectId



The screenshot shows the DataLink interface with a search form and a results table. The search form is highlighted with a pink border and contains the following fields:

- table:
- id_column:
- join_time_column:
- band:
- detail:

The results table below the search form has the following columns: coord dec (deg) double, coord ra (deg) double, and a table with columns: isSource, detect, isIsolated, detect_isPatchInner, and detect. The table contains several rows of data, with the first row highlighted in yellow.

coord dec (deg) double	coord ra (deg) double	isSource	detect	isIsolated	detect_isPatchInner	detect
-35.7991382	62.1479					
-35.800445	62.1464					
-35.7989787	62.1506					
-35.798894	62.1465					
-35.7979753	62.1471					
-35.8003318	62.1453					
-35.798638	62.1506					
-35.7986856	62.1446					
-35.7975025	62.1476873	0	false	true	true	true

- 1) Perform a search in the Rubin “Object” catalog
- 2) New tab in viewer provides auto-generated UI for light-curve query based on the selected row in the results table

RSP TAP Search External Images External Catalogs Add Chart Upload Background Monitor Logout

Data Product: dp02_dc2_catalogs... Coverage

More Pin Table Redo Search Table Chart

VOTable (10 cols x 56 rows)

expMidptMJD (d) double	objectId long	band char	psfFlux (nJy) double	psfFluxErr (nJy) double	psfDiffFlux (nJy) double	psfDiffFluxErr (nJy) double
61084.1099662	1651589610221899038	g	207031.4740315	436.9412266	56304.1613213	438.8301172
61084.0890902	1651589610221899038	g	169028.2265955	395.6898449	18626.9871368	399.099745
60934.2566012	1651589610221899038	g	121269.3527154	334.800716	-28310.9160528	327.063223

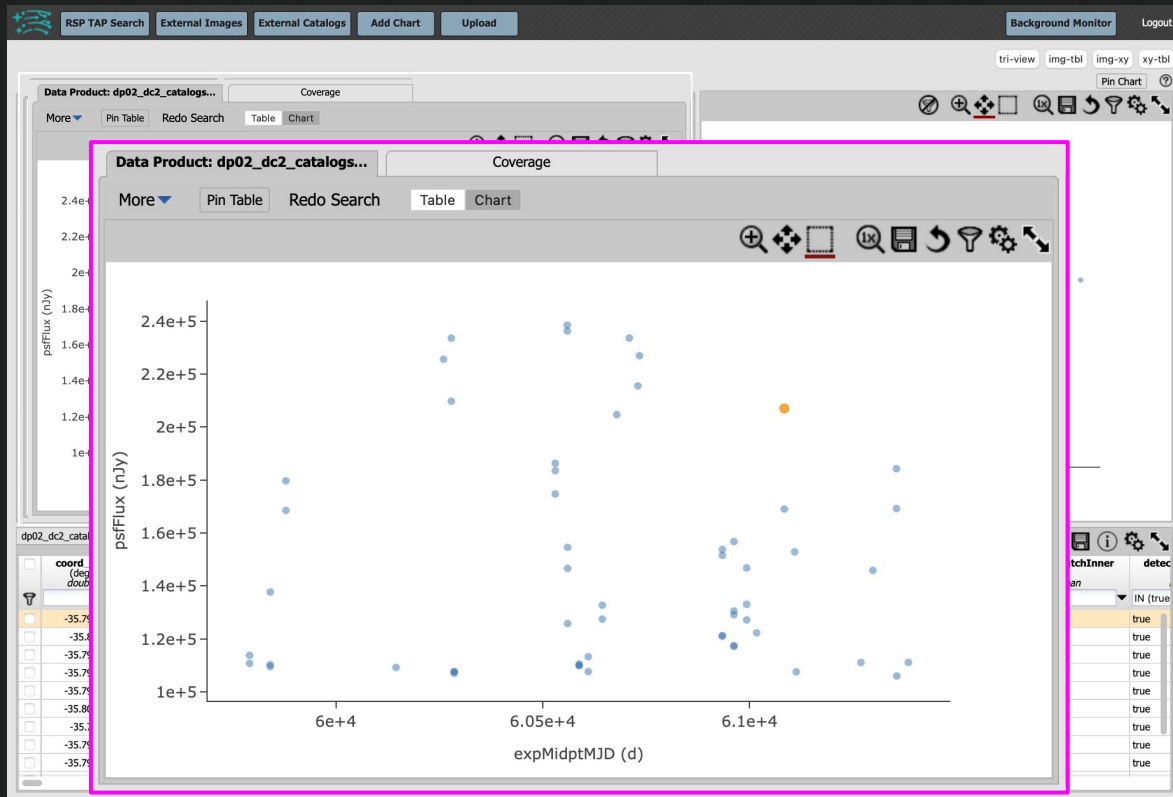
expMidptMJD (d) double objectId long band char psfFlux (nJy) double psfFluxErr (nJy) double

61084.1099662	1651589610221899038	g	207031.4740315	436.9412266
61084.0890902	1651589610221899038	g	169028.2265955	395.6898449
60934.2566012	1651589610221899038	g	121269.3527154	334.800716

dp02_dc2_catalogs.Object - ... x

coord_dec (deg) double	coord_ra (deg) double	deblend_nChild int	deblend_skipped boolean	detect_fromBlend boolean	detect_isDeblendedModelSource boolean	detect_isDeblendedSource boolean	detect_isIsolated boolean	detect_isPatchInner boolean	detect IN (true)
-35.7991382	62.1479018	0	false	true	true	true	false	true	true
-35.800445	62.1464835	0	false	true	true	true	false	true	true
-35.7989787	62.1506534	0	false	true	true	true	false	true	true
-35.798894	62.1465273	0	false	true	true	true	false	true	true
-35.7979753	62.1471533	0	false	true	true	true	false	true	true
-35.8003318	62.1453005	0	false	true	true	true	false	true	true
-35.798638	62.1506054	0	false	true	true	true	false	true	true
-35.7986856	62.1446834	0	false	true	true	true	false	true	true
-35.7975025	62.1476873	0	false	true	true	true	false	true	true

- 1) Perform a search in the Rubin “Object” catalog
- 2) New tab in viewer provides auto-generated UI for light-curve query
- 3) “Submit” yields a **table**...



- 1) Perform a search in the Rubin “Object” catalog
- 2) New tab in viewer provides auto-generated UI for light-curve query
- 3) “Submit” yields a table and a corresponding plot

Additional DataLink services will help users locate many kinds of related data

- Difference-imaging light curves
- Images on which an Object, or a transient, were detected
- Input images to a coadd
- Calibration data associated with the processing of an image
- Reconstructed PSFs at locations of detections

Each new feature requires:

- A simple addition to the TAP service configuration text files
- A corresponding (micro)service to return the data

Our IVOA service framework and “phalanx” Gitops-based deployment system make these easy to roll out incrementally.

New user-visible features appear without requiring changes to the RSP Portal (Firefly) application.

Rubin's VO Implementation experience

Good:

- Yay standards
- Standards documents are generally high quality
- Being able to reuse service code (CADC TAP)
- DataLink offers a great abstraction layer to both the data and generic UIs such as our Firefly-based RSP Portal

Could be better:

- XML rather than JSON
- Case-insensitive parameters
- Error reporting relatively underspecified
- Mixing GET and POST parameters in the same request
- Web Security concerns

Overall, developer experience is too different from current web service patterns and frameworks and is awkward to implement

We'd like to talk about these more in detail and discuss options at the next IVOA InterOp

Documentation:

- Overall platform: phalanx.lsst.io
- Authentication: gafaelfawr.lsst.io

Implementation tech notes:

- Image cutouts: dmtn-208.lsst.io
- More SODA notes: sqr-063.lsst.io
- HiPS: dmtn-230.lsst.io
- DataLink: dmtn-238.lsst.io
- Authentication: dmtn-234.lsst.io

Repositories:

- RSP overall: github.com/lsst-sqre/phalanx
- Image cutouts: github.com/lsst-sqre/vo-cutouts
- HiPS server: github.com/lsst-sqre/crawlspace
- DataLink service: github.com/lsst-sqre/datalinker
- Authentication: github.com/lsst-sqre/gafaelfawr
- Schemas: github.com/lsst/sdm_schemas