

ObsLocTap: Publishing the Rubin Observing Schedule

William O'Mullane

2024-01-25

1 Introduction

Vera C. Rubin Observatory is required to publish the observing schedule nightly. Following our VO first policy, we choose ObsLocTap (Salgado et al., 2021)¹ as the protocol.

This short document discusses some design details.

2 Requirements

LSE-30 (OSS) requires publication at least two hours ahead of observing.

LSE-61 (DMSR) has DMS-REQ-0353 Specification: A service shall be provided to publish to the community the next visit location and the predicted visit schedule provided by the OCS. This service shall consist of both a web page for human inspection and a web API to allow automated tools to respond promptly.

In DMTN-199 the agencies insist we publish a schedule 24 hours ahead of observing. The lack of fidelity of such a schedule is acknowledged by the agencies.

Though an official LCR should be made it is clear we must provide the schedule 24 hours in advance.

3 Design

There are three parts to this:

1. The forecast which comes from the scheduler.

¹<https://www.ivoa.net/documents/ObsLocTAP/>

2. The history of which observations were made which we will need to construct.
3. Publishing compliant with the IVOA ObsLocTap spec (Salgado et al., 2021).

3.1 Scheduler forecast

The scheduler puts out an event record with a number of observations in it². Currently the requirement is 2 hours and the system is built around that kind of window. The observing conditions and schedule are felt to be relatively stable on that time frame. But let us assume there will be a record for 24 hours.³ This shorter record may be produced as frequently as every observation is made with an updated schedule depending on weather etc. Basically when the scheduler has an update a new event is published to SAL.

These events are available at the USDF in the EFD and the last may be found with this influx query:

```
SELECT * FROM "efd"."autogen"."lsst.sal.Scheduler.logevent_predictedSchedule
      where numberOfTargets > 0 ORDER BY DESC LIMIT 1
```

The current structure of this record is a bunch of columnar arrays representing a time and pointing with the values: `mjd`, `ra`, `dec`, `rotSkyPos`, `nexp`. It is understood more values may follow such as `field/tile` and `filter`. The potential mapping of these values to the ObsLocTap Characterization is given in Table 1.

It seems a separate 24 hour schedule produced each night is preferable to the scheduler team. This makes the forecast easy. There would then also be several shorter predictedSchedule events to deal with.

We are required to give 24 hours notice but that is not yet in the baseline requirements.

3.1.1 Handling the schedule

We will need to look at each schedule event record in the EFD as it arrives. For our purposes either polling each 30s or having an influx callback on the EFD at USDF would work.

²<https://ts-scheduler.lsst.io/developer-guide/developer-guide.html#operation-modes>

³At 24 hours before we would, at the start of the observing night, need to produce the schedule *for the next night*.

An initial prototype could simply publish this to a web page or JSON file. This may be a useful tool in any case⁴.

For ObsLocTap a separate record must be created for each pointing in a database lets call this the ObsLocTAP database in this document. ObsLocTap refers to such an entry as a `Characterization`. Putting this in a postgres database would seem feasible and allow a fairly standard TAP implementation to be deployed atop. Postgress is already available at USDF. IVOA ObsLocTap considers a table called `obsplan` and the schema is fixed in the spec (reproduced in Appendix A). The schema shall be defined in the schema repository⁵ allowing us to generate the SQL from the description in the repo.

Based on the required schema of ObsLocTap (reproduced in Appendix A our schema and mapping to the fields shown in subsection 3.1 is provided in Table 1.

So let us assume we shall have a postgres table with the IVOA fields as in Table 1. As each new event record is seen the new pointings not already in the database must be added. If there is a 24 hour schedule some of the new pointings will replace existing ones, the existing entries should be marked `unscheduled`. The simple approach we would follow for new `predictedSchedule` events is to `unschedule` any existing `Characterization` in the time span of the new event.

3.2 History of observations

Next we need to record which observations were made to fill the `ObsDataSet`. There is no ID in the scheduler forecast so there is no explicit matching of observation to forecast. There is no known plan to add such an ID. For every observation made we shall have to do either:

1. If there is a record in the ObsLocTAP database for this pointing mark it as done (set `execution_status=Performed`). We may want to duplicate more than the time to this table but the exposure ID allows one to get any info we needed from the butler registry. Initially at least we will stick with the schema in the ObsLocTap spec.
2. If there is no record create a record with all the forecast field blank.⁶ If there is a record around this time but not this pointing/filter mark that one `execution_status=Unscheduled`

⁴A start has been made in <https://github.com/lstt-dm/obsloctap>

⁵SDM Schema repo https://github.com/lstt-sdm_schemas/tree/main/yml

⁶It is not clear the ObsLocTap requires this.

| Column Name | Mapping to Rubin |
|------------------|---|
| t_planning | logevent_predictedSchedule.mjd |
| target_name | this will always be empty |
| obs_id | dataId['exposure'] or obsid from camera - initially some random number |
| obs_collection | |
| s_ra | ra |
| s_dec | dec |
| s_fov | 3 |
| s_region | we could do this though not sure we should store it |
| s_resolution | 0.2 arcsec |
| t_min | dimensionRecord.timespan.start |
| t_max | dimensionRecord.timespan.end |
| t_exptime | t_max - t_min |
| t_resolution | 15s |
| em_min | Start in spectral coordinates - filter low edge in meters |
| em_max | Stop in spectral coordinates - filter high edge in meters |
| em_res_power | |
| o_ucd | phot.flux.density? |
| pol_states | NULL |
| pol_xel | 0 |
| facility_name | Vera C. Rubin Observatory |
| instrument_name | dataId['instrument'] |
| t_plan_exptime | logevent_predictedSchedule.mjd |
| category | Fixed |
| priority | 1 |
| execution_status | One of the following values: Scheduled, Unscheduled, Performed, Aborted |
| tracking_type | Sidereal |

TABLE 1: IVOA Obsplan columns, proposed mapping to Rubin metadata

This table would now overlap the exposure log but it is not obvious we should combine these as they serve slightly different purposes. It is further noted by the scheduler team that very few of the scheduled observations would be made as planned. The

3.2.1 Was the observation made ?

There is probably no clear cut answer to this given that there is no ID associated with the forecast. If we assume users of ObsLocTap cared about co-observing then their object could be anywhere in the $9deg^2$ area around the pointing. For a first stab lets say the observation was made if all the following are true :

1. The filter used matched the filter in the forecast. This would seem to fairly important but perhaps it is not - I am unsure of the probability of doing the scheduled observation with the wrong filter. It seems unlikely.
2. The ra, dec of the forecast matches (the commanded position should match).
3. The shutter close time of the observation was within 30s of the mjd of the forecast. We are unlikely to be exactly on time so one exposure error seems reasonable.

This is somewhat arbitrary but at least provides a codeable starting point for discussion.

3.2.2 Finding the observations

The butler at the USDF is ingesting the images from the summit within a minute or so. The butler registry can then provide the metadata needed for subsection 3.2.1 Again whether this is polling or call back should be discussed. Some assistance from a butler guru will also be needed to work out an efficient query. The prompt processing is trigger at USDF would provide the correct event for doing this for example since it fires on each new image.

3.3 Deployment

It seems the USDF is the correct place for a small process to run which populates the ObsLoc-TAP database and exposes the obsplan table for the TAP service (subsection 3.4). This should

be deployed with Phalanx⁷.

3.4 Publishing

Assuming we constructed and populate the obsplan table in the previous sections publishing means having a TAP service for that table. We already use CADC TAP on Qserv and there is a version for Postgress so this step may be fairly straight forward.

Once the TAP service is available then it must be registered. This is describe in section 5 of the spec (Salgado et al., 2021).

4 Implementation

There are two parts to this implementation:

1. The postgress database and populating it.
2. Serving that up via TAP.

No 2. here is well understood - we have TAP services so we put one in front of the ObsLocTAP database

There are some choices to be made concerning No. 1.

Frossie provided the simple diagram Figure 1 which captures the ideas.

4.1 Create the table

The yaml was added to sdm_schema so Felis may be used to create the schema.

I did not find Felis on pips so I grabbed `git@github.com:lsst/felis.git` and installed it with `pip install ..`

⁷<https://phalanx.lsst.io>

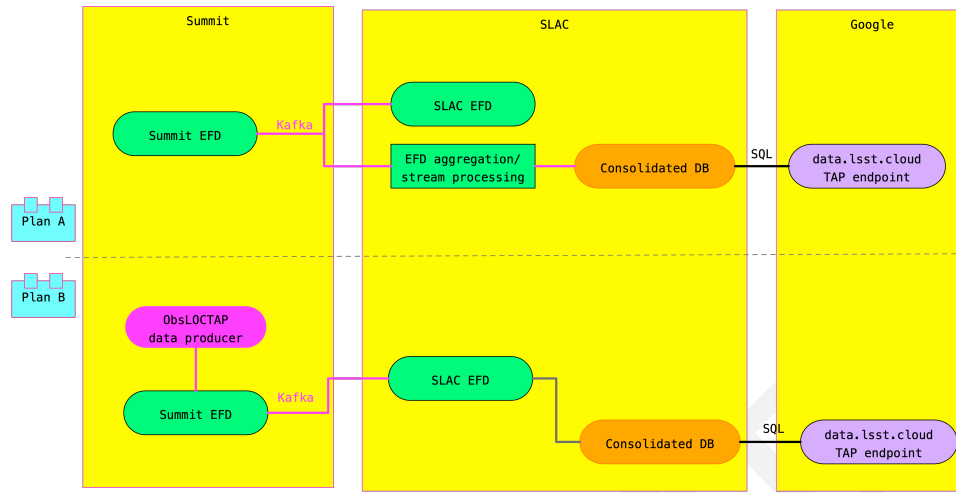


FIGURE 1: Choices on how to populate the postgres database for ObsLocTap.

I also cloned `git@github.com:lsst/sdm_schemas.git`.

This allowed the Felis execution, database login and table creation:⁸

```
Felis create-all --engine-url="$engine_url" --dry-run sdm_schemas/yml/obsloctap.yaml > obsloctap.sql
```

```
psql -h usdf-butler-session.slac.stanford.edu -U rubin lsstdb1
```

```
lsstdb1=> create schema obsloctap; lsstdb1=> create schema obsloctap_dev;
```

```
SET SEARCH_PATH = obsloctap_dev;
i obsplan.sql SET SEARCH_PATH = obsloctap;
i obsplan.sql
```

Can check the default schema and if it exists with: `show SEARCH_PATH;`
`dt`

⁸ivoa. was removed from the DDL

4.2 Populating the ObsLocTAP database

Initially at least we should try plan A and do all of this at the USDF. The EFD messages are replicated there so we can pick up the scheduler messages and process them via a stream processor in Kafka (within Sasquatch).

The 24 hour schedule should probably get a new topic to make it distinct from the normal operations - this has to be agreed with scheduler/summit. This will be processed to create schedule entries.

Then we will process `1sst.sal.Scheduler.logevent_predictedSchedule` to update/create schedule entries.

Finally we will look at `1sst.sal.ATHeaderService.logevent_largeFileObjectAvailable` which will give us a pointer to the header (url field) and using `astro_metadata_translator` we can get `r_obsid`, `exposure_id`, `filter` and update the ObsLocTAP database appropriately.

4.3 TAP service

There is already a ticket DM-39729 for the creation of the Felis schema. Another is needed to expose this via tap.

Along the lines of separation of security concerns this would be deployed on the Cloud (US DAC) not in USDF.

Currently the endpoint `/schedule` will return the last 1K lines from the Obsplan table as a json file. This is deployed only on `usdfdev` as of Dec 2023. <https://usdf-rsp-dev.slac.stanford.edu/obsloctap/schedule> It only contains test rows.

4.4 Phalanx

Deployment is with Phalanx see <https://phalanx.lsst.io/index.html>. Specifically <https://phalanx.lsst.io/applications/obsloctap/index.html>.

A Obsplan table from ObsLocTap

The obsplan tabel schema from the "Observation Locator Table Access Protocol Version 1.0" is provided here for ease of reference.

Draft

| Column Name | Description | Constraint |
|------------------|--|--|
| t_planning | Time in MJD when this observation has been added or modified into the planning log | not null |
| target_name | Astronomical object observed, if any | |
| obs_id | Observation ID | not null |
| obs_collection | Name of the data collection | |
| s_ra | Central right ascension, ICRS | |
| s_dec | Central declination, ICRS | |
| s_fov | Diameter (bounds) of the covered region | |
| s_region | Sky region covered by the data product (expressed in ICRS frame) | |
| s_resolution | Spatial resolution of data as FWHM | |
| t_min | Start time in MJD | not null for execution_status Scheduled or Performed |
| t_max | Stop time in MJD | not null for execution_status Scheduled or Performed |
| t_exptime | Total exposure time | not null for execution_status Scheduled or Performed |
| t_resolution | Temporal resolution FWHM | |
| em_min | Start in spectral coordinates | |
| em_max | Stop in spectral coordinates | |
| em_res_power | Spectral resolving power | |
| o_ucd | UCD of observable (e.g., phot.flux.density, phot.count, etc.) | |
| pol_states | List of polarization states or NULL if not applicable | |
| pol_xel | Number of polarization samples | |
| facility_name | Name of the facility used for this observation | not null |
| instrument_name | Name of the instrument used for this observation | |
| t_plan_exptime | Planned or scheduled exposure time | |
| category | Observation category. One of the following values: Fixed, Coordinated, Window, Other | not null |
| priority | Priority level { 0, 1, 2} | not null |
| execution_status | One of the following values: Planned, Scheduled, Unscheduled, Performed, Aborted | not null |
| tracking_type | One of the following values: Sidereal, Solar-system-object-tracking, Fixed-az-el-transit | not null |

TABLE 2: ObsPlan columns description

B References

[LSE-30], Claver, C.F., The LSST Systems Engineering Integrated Project Team, 2018, *Observatory System Specifications (OSS)*, LSE-30, URL <https://ls.st/LSE-30>

[LSE-61], Dubois-Felsmann, G., Jenness, T., 2019, *Data Management System (DMS) Requirements*, LSE-61, URL <https://lse-61.lsst.io/>, Vera C. Rubin Observatory

[DMTN-199], O'Mullane, W., Allbery, R., AlSayyad, Y., et al., 2023, *Rubin Observatory Data Security Standards Implementation*, DMTN-199, URL <https://dmtn-199.lsst.io/>, Vera C. Rubin Observatory Data Management Technical Note

Salgado, J., Ibarra, A., Ehle, M., et al., 2021, Observation Locator Table Access Protocol Version 1.0, IVOA Recommendation 24 July 2021, ADS Link

C Acronyms

| Acronym | Description |
|---------|--|
| API | Application Programming Interface |
| CADC | Canadian Astronomy Data Centre |
| DAC | Data Access Center |
| DB | DataBase |
| DESC | Dark Energy Science Collaboration |
| DM | Data Management |
| DMS | Data Management Subsystem |
| DMS-REQ | Data Management System Requirements prefix |
| DMSR | DM System Requirements; LSE-61 |
| DMTN | DM Technical Note |
| EFD | Engineering and Facility Database |
| FWHM | Full Width at Half-Maximum |
| ICRS | International Celestial Reference Frame |
| IVOA | International Virtual-Observatory Alliance |

| | |
|-----------|---|
| JSON | JavaScript Object Notation |
| LCR | LSST Change Request |
| LSE | LSST Systems Engineering (Document Handle) |
| MJD | Modified Julian Date (to be avoided; see also JD) |
| OCS | Observatory Control System |
| OSS | Observatory System Specifications; LSE-30 |
| ObsLocTAP | Observation Locator Table Access Protocol (IVOA standard) |
| SAL | Service Abstraction Layer |
| SQL | Structured Query Language |
| TAP | Table Access Protocol (IVOA standard) |
| UCD | Unified Content Descriptor (IVOA standard) |
| US | United States |
| USDF | United States Data Facility |
| VO | Virtual Observatory |