



Image Service Architecture

Gregory Dubois-Felsmann
Caltech/IPAC

DMLT Virtual meeting – 23-24 February 2021

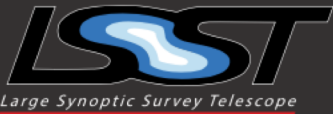




Outline

1. Creating Image Metadata
2. Serving Image Metadata
3. Image-related Services

DMTN-139 – Image Service Architecture



DMTN-139: LSST Image Service Architecture

Edition: tickets/DM-22746

Switch editions

Edit on GitHub

- 1 Introduction
- 2 Requirements
 - 2.1 SRD (LPM-17)
 - 2.2 LSR (LSE-29)
 - 2.3 OSS (LSE-30)
 - 2.4 DMSR (LSE-61)
 - 2.5 LSP Requirements (LDM-554)
- 3 Planned Image Data Products
 - 3.1 Deep Coadds
 - 3.2 Single-epoch Images
- 4 Use Cases

DMTN-139: LSST Image Service Architecture

Gregory Dubois-Felsmann

Latest Revision: [2020-10-21](#)

Note

This technote is not yet published.

This document describes the architecture for Internet-visible image services to be provided by LSST. It explains the use of IVOA standards to provide these services, and discusses where LSST's needs may require extensions to those standards. It addresses the back-end architecture that supports these services, including especially the role of the Gen3 Butler.

Image services include: observation metadata services based on ObsCore and CAOM2, image retrieval services, image cutout and mosaicing services, image-recreation-on-demand, and HiPS multi-resolution image map services.

This note discusses the role of the image services in the LSST Science Platform and outlines expectations for how they will be used internally and externally.

Later editions of this note may discuss issues associated with IVOA Registry representation of the services, or they may be moved to a separate technote on our Registry interface.

1 Introduction

- 4 Use Cases
 - 4.1 Image Cutouts
- 5 Community Standards
 - 5.1 IVOA
 - 5.2 CAOM2
 - 5.3 FITS
- 6 Service Concept
 - 6.1 Image Data Formats
 - 6.2 Observation Metadata
 - 6.3 Special Case: HiPS
- 7 Service Details
 - 7.1 ObsTAP Service
 - 7.2 TAP Service for Additional Observation Metadata
 - 7.3 SIAv2 Service
 - 7.4 SODA Service(s)
 - 7.5 HiPS Service
 - 7.6 Availability Endpoints
- 8 User Access Scenarios
 - 8.1 Portal Aspect
 - 8.2 Notebook Aspect and External Access
- 9 Major Open Questions
 - 9.1 IVOA Registry Representation
 - 9.2 HiPS Discovery
 - 9.3 MOC Discovery
- 10 Appendices
 - 10.1 Appendix: Implementation Notes
 - 10.2 Appendix: Portal-derived Requirements on Image Services
 - 10.3 Requirements
 - 10.4 Service Implications



Types of image metadata

- Gen3 Butler repository data
- Community-standard metadata
 - ObsCore
 - CAOM2
- “Original concept” / “cat package” tables (“Visit”, “Exposure”, etc.)
- Actual PipelineTask outputs

Converting image metadata

- Community-standard metadata story is in fairly good shape
 - Substantial progress on Gen3-to-ObsCore
 - Mandatory and some “optional” ObsCore metadata satisfied
 - Able to perform conversion without opening image files
 - Much ObsCore metadata obtained from lookup based on Butler DatasetType
 - Dataproduct_type, CalibLevel, UCDs, pixel scale, etc.
 - Handles both single-epoch and coadded image metadata
 - No recent work on CAOM2, but...
 - Basic framework was worked out long ago
 - Much of the substance of CAOM2 is common with ObsCore; expect this part to be pretty straightforward work
 - Ultimately the primary workflow is Gen3-to-CAOM2; ObsCore can be generated as a view
 - CAOM2 provides for a limited view of image provenance (specifically, things like visit-to-coadd)
 - This is going to need a close look in the next quarter
 - We have all the data, just need to figure out the flow for where to do the “ETL”

“Original concept” metadata

- Our data model has had Visit, Exposure, etc. tables since the dawn of time
 - Much Rubin/LSST-specific information, including measured performance (which is only treated in a very limited way in ObsCore/CAOM2)
 - The necessary data appears scattered across the Gen3 repository and the outputs of various PipelineTasks
- The most obvious stand-alone approach for creating these tables would be to do “SDM Standardization”
 - We need to take a look at this workflow to see how to lay it out, but most likely the present scheme could be adopted.
- One more thing...

Too many types

- So far we end up with Gen3, CAOM2 (and its ObsCore view), and the “original” tables – confusing to users and blurs what is authoritative information vs. what is a spray-on compatibility layer
- We decided after careful thought ~2 years ago that CAOM2 could not be used for the data model for Gen3 – too many compromises; not seen even by its originators as a “working” data model for production
- Can we use CAOM2 – extended with additional data – as a replacement/vehicle for the data from the “original concept”?
 - I think there’s a good chance of it
 - gpdf, Colin, Yusra have agreed to look at this – should have started before this meeting
 - If successful, we would end up with essentially only two data models

Serving image metadata

- Threefold baseline:
 - Direct Butler queries
 - At least in Notebook Aspect
 - Recent work leading to making this possible remotely via API Aspect and thin Python layer
 - IVOA image metadata services
 - ObsTAP (“ivoa.ObsCore” table in a TAP service)
 - SIAv2 (shares data model with ObsCore, implementable as a layer over ObsTAP; CADC Java)
 - Database queries via TAP against detailed tables
 - CAOM2 tables following the community-standard data model
 - And either (see above)...
 - Our own “original concept” tables
 - Our project-specific data integrated into the CAOM2 structure

Metadata creation and loading workflows

- Scenarios
 - Data Releases
 - Standard nightly observing
 - Commissioning / V&V
- Workflow for Data Releases is mostly straightforward
 - Natural role for “afterburners” that standardize data
 - Need to think about whether anything other than Gen3 and PipelineTask outputs is available internally in the long course of the creation of a DR – i.e., will we want to use tooling based on (subsets of) the standardized metadata internally?
- However...

Workflows closer-to-real-time

- Nightly data-taking during operations
 - Baseline 6-to-24-hour latency for data-rights-user data access
 - Need to support queries against standardized metadata on this time scale
 - 6-hour latency implies sub-night data release units
 - Atomic release of an entire night's data, during the daytime, is a bit simpler
 - How do we a) generate this data and b) load it into a queryable service?
 - Present ObsTAP prototype is loaded via a fairly static process; not clear that it scales to a 7-day-a-week process, let alone a sub-night one
 - Rapid release has to take referential integrity into account – when the metadata record is released, the image itself has to be available and any linked services (e.g., cutouts) need to be available

“Internal” workflows

- During commissioning (also applies to normal ops)...
 - Are image metadata services updated “continuously” during a night, at least for raws?
 - Or is Butler access the only near-real-time service?
 - Strong reasons to think an ops team might want Web-based services
- During V&V and commissioning...
 - Will we make “mini-releases” available through the image metadata and image services?
 - E.g., monthly HSC processings, rapid-turnaround reprocessings of early ComCam data, etc.
- Let’s see if we can architect a workflow that can support more timely use cases

IVOA image metadata services

- ObsTAP and SIAv2 are both based on ObsCore
 - ObsTAP puts **ALL** image metadata into a **single table** per TAP service
 - Different “sets” of data from a single facility/telescope are distinguished primarily by “Collection” – NB that this is not necessarily 1:1 with Gen3 collections.
 - We have not yet worked out the full complexity of the mapping to the ObsCore obs_collection values
 - The same applies to an SIAv2 service – it’s one big bag of data records
- Query mechanisms:
 - ObsTAP queries are done with ADQL constructs, e.g., CONTAINS(), INTERSECTS()
 - SIAv2 is done with a standardized set of URL query parameters (e.g., “&POS=CIRCLE 20.71 -0.24 0.01”)
 - Both return results as VOTables of ObsCore records

IVOA image access basics

- ObsCore query results include:
 - access_url – a URL usable with GET
 - content_type – a MIME type
- The URL can either return the image directly, or, more usefully...
- The URL can point at a DataLink “links service” which itself returns a VOTable of URLs to the image and to related data and services, e.g.,
 - Primary image
 - Compressed image
 - Thumbnail
 - Cutout service
 - Image re-creation service
 - Provenance queries
 - Query for sources in the image

What do the static GET URLs for images get?

We need to reply to queries with FITS files (other formats can be supported, too)

- Static files (yes, they exist)
 - Thin service over the IDF/USDF object store
 - Are we OK with serving static files? It was not in the original FDR-era LSST concept.
 - Serving static files is much more scalable to peaks in load (CDN)
 - Just the CCD-scale compressed PVI and the patch-scale coadd tiles?
 - *Considering also serving (somewhat) smaller tiles to facilitate a large subset of end-user queries*
- Active services (must be low-latency) that always create files on demand
 - Original “imgserv” design
 - Allows metadata to be updated to “best available” – e.g., applying final WCS to raw images
- Initial services will use the already-available static files
- Aim is to use direct URLs to the object store – is this in fact workable from an access-control perspective?

Non-IVOA access

- Not much to say here...
- Plan is to make CAOM2 tables available in TAP, along with linked (JOINable) tables with Rubin-specific attributes
 - Or a separate set of Rubin-specific tables if we can't make that work. :(
 -
- CAOM2 uses the same `access_url` notion as ObsCore, no extra work there
- Use TAP_SCHEMA foreign-key definitions and DataLink service descriptor records in the TAP service to document the links between tables

Work needed on TAP (and its back end)

- Architecture is strongly dependent on the TAP service being able to annotate its outputs with IVOA-standard DataLink records
- CADC TAP has a very limited undocumented capability for this
- IRSA TAP (C++ and not realistically usable outside IRSA) has a full implementation of this which has been shown at IVOA and can be used as a model; it would completely cover our needs
 - CADC is interested in moving in this direction and we should collaborate
- Also need to understand how to rapidly deploy new data into the ObsTAP service
 - Remember that it's a single table! (Or a view of a join on 3 equally monolithic CAOM2 tables.)
 - If ObsTAP is a view, does it need to be materialized? How often?

SIAv2

- CADC has a Java implementation of SIAv2 as a layer over ObsTAP
- We should evaluate this to see if it works for us
- The architecture prioritizes ObsTAP but we should definitely support SIAv2 in the end
 - Firefly-based interim Portal will use ObsTAP for image queries
- Image access once you have the metadata output works identically to the way it does for ObsTAP

Sidebar: Firefly status

- Image browser based on a table ObsCore records returned from a query has been working since before the “freeze”
- ObsTAP services can be queried today but there is very limited UI support for constructing queries against ObsCore-specific concepts (e.g., you have to know the magic `calib_level` values)
- Brian Van Klaveren is actively adding ObsCore-specific UI elements to the query screen
 - Initial UI has been laid out; working on fleshing out the actual behavior

Links service

- This is a very simple service
- Takes image ID (obs_publisher_id) as input
- Returns table of links as output

- We need to define and assign this work
 - The architecture document will include an initial spec
- Extensible over time to include more and more related services
- Client software (both UIs and Python libraries) can and does use these links to provide a good experience
 - E.g., Firefly will be able to allow users to follow links from a coadded image to its input images

Associated data services

- Links services (and DataLink in general) work best with a set of associated micro-services that support the onward queries
- Examples:
 - Tiny redirection service that converts a URL query for the sources in an identified image to the appropriate TAP query
 - Service to return a VOTable of ObsCore data for the inputs to a coadd, based on the coadd image ID



Associated image services

- Services for cutouts and for re-creation of “virtual data products”
- Working on an architecture that facilitates deploying such services based on PipelineTask pipelines
- URL parameters converted to ad-hoc objects in the Butler data model
- Build the framework once, become able to deploy whatever services are needed
- Need a UWS-to-IDF/USDF-batch gateway
 - Related to OCPS work

Significant development tasks

- Metadata-conversion workflows
- Links service implementation
- UWS server framework for PipelineTask pipelines
 - Parts of this emerging from current cutout service work
- Framework for redirecting simple URL queries to TAP queries
 - Conceivable to do with rewrite rules, no active server needed
- TAP support for DataLink
- Specific services
 - Authenticated endpoints for statically-held images
 - Cutouts (well under way)
 - Need to add support for the “Ciardi usecase” of requesting the full time series of (compressed) PVI cutouts for a sky location
 - Specific re-creation services
 - PVIs
 - Diffims
 - Alternate coadds
- PVI-compression workflow



Document status

- Will move initial version to master this week
- Refinements will still be needed on metadata unification and conversion workflow decisions