LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Science Platform Design

Gregory Dubois-Felsmann, Frossie Economou, and Kian-Tat Lim

LDM-542

Latest Revision: 2017-07-03

Abstract

This document describes the design of the LSST Science Platform, the primary user-facing interface of the LSST Data Management System.

Change Record

Version	Date	Description	Owner name
1	YYY-MM-DD	Unreleased.	Gregory Dubois-Felsmann, K-T Lim

Contents

Science Platform Design

1 Introduction

1.1 The Portal Aspect

1.2 The Notebook Aspect

1.3 The API Aspect

2 The Architecture of the Science Platform

This section presents the overall architecture of the LSST Science Platform (LSP, not to be confused with the Science Pipelines "stack" code), including its major components and interfaces and design principles.

2.1 Design Overview

The LSP is a multi-tier architecture with distinct but inter-related user-facing "Aspects" over an underlying layer of services that in turn rests on computational infrastructure.

2.1.1 Functional Architecture

The Portal Aspect, JupyterLab Aspect, and Web APIs Aspects provide three distinct user interfaces. Internally, the LSP interfaces with the Data Backbone via the Data Butler client interface and direct "native" Data Backbone interfaces to provide access to the Science Data Archive and (for some instances) other data products. The LSP also has its own data storage for Portal configurations, user notebooks, and other user files and databases. It interfaces with compute provisioning services, authentication and authorization services, and resource management, including a proposal management system to handle requests for resources. Networking connects the LSP with internal systems and the external world and provides a measure of cybersecurity.

The Portal Aspect is implemented using the language-agnostic Web APIs to retrieve data. The JupyterLab Aspect can also use the Web APIs if the user desires, either through a VO client or directly as a Web service.

2.1.2 Deployment Architecture

The LSP is designed to be deployable in multiple locations within the LSST DMS. The production instances are the US Data Access Center, the Chilean Data Access Center, the Science Validation instance in the NCSA Enclave, and the Commissioning Cluster instance in the Base Enclave. Additional instances are used for integration testing in the Integration environment and for science pipelines developer usage in the Developer environment. LSP developers will of course instantiate components of the LSP during their own development.

The LSP is composed of multiple services that are containerized and deployed using Kubernetes. The underlying provisioning of compute resources is handled by the Provisioning and Deployment Service component of the LSST DMS. The technology to be used for provisioning in this component is being evaluated and may vary between instances. vSphere, HTCondor, SLURM, and OpenStack are all potential alternatives. Provisioning is elastic: JupyterLab dynamically instantiates notebooks on demand, and the Portal and Web API services can be expanded or contracted as needed. This elasticity and fault tolerance considerations require that the services hold minimal non-persistent state so that loss of a single service instance has correspondingly minimal impact on users.

2.2 Data Access and Storage

Since the goal of the LSP is to retrieve and analyze LSST data products, data access and the storage of user data are key aspects of the platform.

2.2.1 Databases

Since the LSST catalog data products represent the best measurements of astrophysical phenomena available to the project as well as metadata describing how data was taken and processed, it is anticipated that most uses of the LSP will involve these catalogs which are stored in databases.

2.2.1.1 Database content overview Several types of databases will be accessible to the LSP instances:

- Image and visit metadata
- Catalogs
- Composite data (binary data in catalogs)
- Observatory metadata (including EFD)
- Reference catalogs

Image and visit metadata includes information about the observation captured at the time. It also includes information derived later from analyzing the data and metadata, such as an accurate mapping from pixel coordinates in an image to sky coordinates or an estimate of the photometric zeropoint in an image. The metadata will follow the Common Archive Observation Model [1] as its core. This metadata will be stored in relational databases and made available via ADQL queries.

Catalogs are described in detail in the Data Products Definition Document LSE-163. They include summary measurements of astrophysical Objects, plus measurements in images from each epoch of observation and in difference images. These will also be stored in relational databases and made available via ADQL queries. The catalog of all Alerts that have been issued may be stored in a different database technology, such as a document database, and may thus use a different query interface.

The above catalogs will include "BLOB" fields that provide large data items that are useful to retrieve along with other catalog data but are not suitable for searching. Examples include samples of the posterior probability surface for a model, photometric redshift probability distribution functions, or "heavy footprints" giving deblended pixel values for an Object. These are generally stored in separate tables within the relational database as they are used less frequently.

The Engineering and Facility Database, which contains a record of all commands, configuration, and telemetry from the Observatory systems, will be transformed to provide simplified query access to conditions during each observation. All data present in the original EFD will

be available in the transformed EFD with the exception of Internal, Sensitive, or Highly Sensitive information, as defined in the Information Classification Policy LPM-122, which will not be available to users other than project staff. The EFD does not contain Protected User data. The Transformed EFD will be stored in a relational database, accessible via ADQL queries.

Reference catalogs used in the LSST Alert, Calibration Products, and Data Release Productions will be provided, also in relational databases. Additional catalogs, e.g. from precursor or concurrent surveys in a variety of wavelengths, will also be provided as relational databases depending on usefulness and available resources.

2.2.1.2 Database technologies While many of the above databases will be implemented in "conventional" relational database technology, the largest catalogs will require a distributed system to provide sufficient performance. The LSST-developed Qserv distributed database will be used to implement these. The two implementations may vary somewhat in their support of ADQL features. Users creating their own catalogs (see section 2.2.5) will need to specify whether they are to be distributed, thereby selecting the appropriate back-end implementation.

2.2.2 Images

LSST image data products include raw images, processed visit images, difference images, coadded images, and templates. These are part of the Science Image Archive in the Data Backbone. All of these are accessible to the LSP.

Some data products are currently specified to be "virtual": they are regenerated on demand from other persistently-stored data products, reflecting the estimates of the technology-dependent space/time cost trade-off. Such virtual data products are regenerated for the LSP by services that are part of the LSP Web APIs.

Images will be delivered in at least FITS format; other formats may be added. The LSST DM internal image data format is currently FITS as well, but that may change in the future. Any necessary translation between internal format and external format is performed by services that are part of the LSP Web APIs.

Image data products typically contain additional information besides image pixels. This infor-

mation includes metadata describing the conditions under which the image was generated, its provenance and processing history, if any, and metadata derived from the image itself. Much of this metadata can be expressed as simple data types in "headers" or similar mechanisms, but some, such as mask and variance planes, pixel-to-sky mappings, and point spread function (PSF) models, may be too complex for such a representation. When this information is specific to the image, LSST DM generally stores this information in the same file as the image pixels, as additional pixel planes or extension tables. The structure and content of these will be documented for LSP users.

2.2.3 LSST-specific Objects

The Python objects used by the Science Pipelines code do not always correspond one-to-one with persisted catalog or image data products in the Data Backbone. For example, an image and its background model may be persisted in two distinct files. The PSF model associated with the detections in an image may be stored in a file while the information about the detections is stored in a database catalog.

The component files and catalogs are directly accessible via the LSP, but for convenience the Data Butler client library provides mechanisms to define composites that combine multiple components into a single retrieved Python object. Since this usage is specific to the Python language, it is only directly visible to the user in the LSP in the JupyterLab Aspect.

2.2.4 Data Access Services

The LSP Web APIs Aspect provides services to retrieve catalog and image data products. Access to catalog data products is provided by a Web service that supports the VO TAP protocol with queries specified in ADQL. Access to image and file data products is provided by a separate Web service that supports the VO SIA, SODA, and VOSpace protocols. The image and file service is also responsible for regenerating "virtual" data products and converting from internal to external data formats as necessary. Both of these Web services support asynchronous operations, as some requests will have long response latencies (e.g. large-scale catalog queries or data regeneration). Access to metadata about images and files is provided by TAP/ADQL queries on metadata tables, rather than a distinct metadata service. More detail on these services is provided in section 5.

Within the JupyterLab Aspect, and internal to the Web APIs Aspect, the Data Butler client interface may be used to retrieve Python objects, and a "native" Data Backbone API may be used for high-performance retrieval of datasets. The Portal Aspect uses the Web APIs Aspect exclusively for data access; it does not use the Data Butler nor the "native" Data Backbone API.

2.2.5 User Catalog Data Support

Users may create their own catalogs within the LSP, a capability that is patterned after SDSS SkyServer's "MyDB". Catalogs may be derived from queries to the LSST catalogs or loaded from computations or external data sources. As mentioned above in section 2.2.1.2, large distributed catalogs meant for joining with the LSST data products must be identified so that they can be stored appropriately in the Qserv database.

All three Aspects will provide mechanisms for creating user catalogs; the Portal's mechanism will be implemented via the Web APIs, as usual. User catalogs may be accessed in the same ways that LSST catalog data products are accessed.

User catalogs are backed up for disaster recovery purposes, but they are not fully protected against user error.

2.2.6 User Workspace Storage

Users may create their own images and files within the LSP. These all live within a User Workspace that is common across all three Aspects. All three Aspects will provide mechanisms for creating and retrieving user files. In the JupyterLab Aspect, this is via a mounted filesystem. In the Web APIs Aspect, this is via VOSpace or WebDAV.

User file workspaces are backed up for disaster recovery purposes, but they are not fully protected against user error.

2.2.7 Data Access Permissions and Quotas

All LSST data products are available to all users with LSST data rights. Users who had LSST data rights but have since lost them may be granted access to the data products available as

of the time of loss but not newer ones.

Users may grant other users or groups of users access to catalogs and files in their workspace. Such access may be read-only or read-write.

Per-user catalogs and files are a limited resource with quotas determined by resource management (see section 2.5). Some "user" catalogs and files may be considered to be owned by a group, with a distinct quota.

2.2.8 Support of Previous Releases

The LSST baseline currently anticipates that the data products contained in the last two Data Releases (as well as Level 1 Alert Production data products) will be available through the LSST Science Platform. Data products from other Data Releases would need to be retrieved via the Bulk Distribution service; they would not be visible within the LSP.

A request to have the catalogs from all Data Releases be available through the LSP is being evaluated.

2.3 Computing Resources

The LSP provides access to computing resources to enable users to analyze LSST data products, alone and in combination with user-provided data.

2.3.1 Basic user compute services

Users of the Portal Aspect share computing resources allocated to the Portal as a whole. The same strategy is used for the Web APIs. Within those shared resources, per-user accounting is performed for resources such as queries and temporary disk space so that resource management policies may be implemented if required.

Each user of the JupyterLab Aspect is assigned a virtual machine per notebook. The virtual machine can be selected from a menu of preconfigured installations of LSST and other software, or users can customize a virtual machine and save it for future use. The resources allocated to the virtual machine are defined by policy.

2.3.2 Large-scale batch and parallel computing

Users of the LSP (any Aspect) may trigger large-scale, long-latency, asynchronous data processing. This processing is handled by submitting jobs to a batch system based on HTCondor. Mechanisms are provided in the three Aspects to monitor the status of batch jobs and retrieve their results. Instances of the workload and workflow management tools and services used by the LSST Data Facility to manage large-scale productions will be made available to each LSP instance, although these will be separate and distinct from the production instances, of course.

2.4 Authentication and Authorization

Identity management and associated roles and authorizations are provided by a unified LSST system as described in LSE-279. This system allows the use of external identity providers such as the Chilean COFRe federation [2] or the US InCommon [3] or GitHub; those verified external identities are then mapped to internal LSST identities. All uses of all instances of the LSP, including the Web APIs Aspect, are authenticated.

2.5 Resource Management

LSP users in the DACs will receive a certain quota of resources (compute, storage, query, bandwidth, etc.). Usage beyond that quota will be allocated by a committee. Outside the LSP, a proposal management system will track requests and allocations that will then be implemented by per-user resource management features within each LSP component.

2.6 Cybersecurity Considerations

The Data Access Center instances of the LSP are the only ones exposed directly to the public Internet. The Science Validation, Commissioning Cluster, Integration, and Developer instances are only reachable from LSST-internal networks, helping to reduce the attack surface. The DAC instances are isolated (on a different LAN) from the production Level 1 and Level 2 domains within the NCSA enclave. Although they may share underlying computing resources, such resources are provisioned for either production or DAC but not both at the same time. The LSST data products within the Data Backbone will be read-only to the DAC LSP instances. The Data Backbone interfaces are all authenticated, but they will be used for some types

of user data, so the Backbone must be hardened against attack (or accident) from the LSP. Provisioning separate virtual machines per user in the JupyterLab Aspect helps to prevent unwanted sharing of information between users. The Portal Aspect and Web APIs Aspect need to be hardened in the same ways as any Web-enabled service, including sanitizing user inputs, avoiding cross-site scripting, etc.

2.7 Additional Support Services

System management and monitoring services for the LSP will be provided by the infrastructure layer supporting each instance. Monitoring at the system and application level will be exposed to LSP users where consistent with user privacy and system security.

3 The Portal Aspect

3.1 Generic Data Browsing

3.2 Documentation Delivery

3.3 Semantics-aware Workflows

3.4 Use of the LSST Stack

3.5 Extensibility of Visualizations

3.6 Extensibility of the UI

4 The Notebook Aspect

The Science Platform offers in-house scientists and engineers as well as (in operations) general science users the opportunity to interactively explore the LSST dataset from a python notebook environment.

4.1 JupyterHub / JupyterLab service

The notebook service is a scalable kubernetes deployment with JupyterHub spawning individual JupyterLab pods for each user. The basic architecture is outlined in Figure 1. JupyterLab is a rich notebook architecture that offers terminal access and dashboarding in addition to the traditional Jupyter Notebook concept and the use of JupyterHub allows it to be scaled to demand as infrastructure allows.

A user of the JupyterLab sevice will hence be able to interact with LSST services in the same way they might have with python scripts in a shell environment. They can use our python interfaces to the web APIs and the Butler for data access, the interface to the workflow system for batch processing, visualisation via bokeh, Firefly, matplotlib etc.

The capabilities of the service are tied to major project milestones outlined in the table below.

Version	Audience	Target L2 Milestone	Shims ok?
proto	SQuaRE and Arch	none	y
alpha	DM devs, Commissioning team training	LDM-503-1	y
beta	Commissioning Team	LDM-503-9	y
1.0	Commissioning (ComCam) Capabilities	LDM-503-12	n
2.0	Science Verification Capabilities	LDM-503-14	n
3.0	General Science User Capabilities	LDM-503-16	n

Note in particular that in addition to the originally planned “Level 3” science user capabilities, the JL service is the primary interface of the LSST System Scientist to Commissioning. This introduces scope not required by the general science user, eg. timely access to data in the Engineering Facilities Database and the ability to rapidly deploy instances of the service on the Commissioning Cluster.

For analyses requiring significant data processing, a process for transitioning a trial-and-error notebook into a batch job involving execution over a large dataset will be provided.

For detailed architectural discussions of the JupyterLab development work-in-progress, its interfaces to other components and the current schedule for major and internal milestones see SQR-018. As technical choices are baselined they will transition to this document.

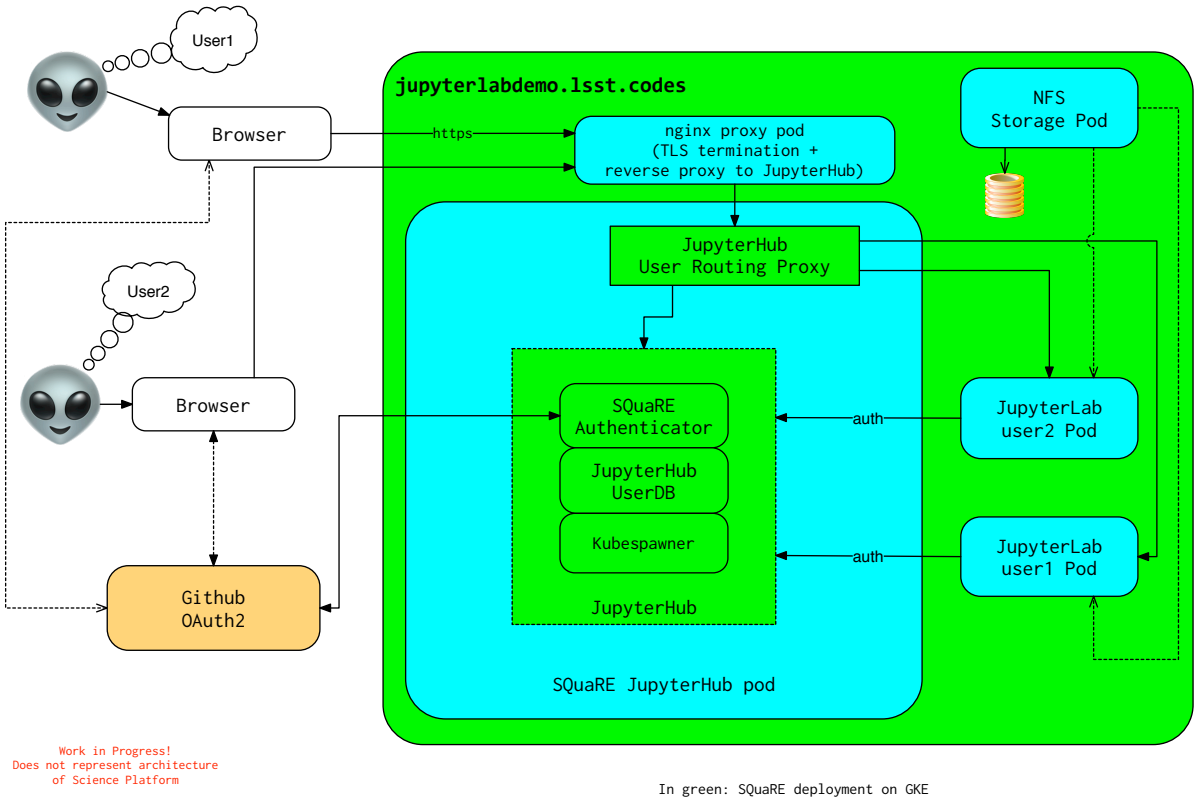

FIGURE 1: Architecture of the proof-of-concept (alpha) JupyterLab deployment

4.2 Pre-installed LSST software

An important advantage of the JupyterLab environment is that it is already set up and optimised for running a variety of LSST data. We achieve this by building containers directly from the weekly releases created by our continuous integration system and then deploying them to JupyterHub which asks the user which version to use to spawn their pod. This means the users always have a choice of cadence (latest weekly or last stable) in their environment depending on their risk tolerance versus appetite for the latest feature. We plan to make other complementary third-party astronomical packages (eg. sextractor) available too.

4.3 Pre-configured access to LSST data

The JupyterLab service will make use of standard LSST stack features such as DAX services, the Butler, and their initialisation through the SuperTask Activator to allow python-level access to LSST data. After that users can use a mix of their own code and stack classes to manipulate and visualise the data. An example notebook running on our prototype deployment using classes from the LSST stack is shown in Figure 2.

5 The API Aspect

The Web APIs provide language-agnostic, location-agnostic authenticated access to LSST data. They are intended to be used by client tools and automated processes. They are not optimized or supported for direct human use via a browser, although they can be used that way. The available endpoints and the URL structures they accept will be documented.

The Web APIs are undergoing detailed design. Initial prototypes exist that provide non-VO-compliant access to catalogs, images, image mosaics, and image cutouts. This section describes the ultimate goals for the LSP Web APIs and documents design features where they are already known.

5.1 Overview - VO services and custom LSST services

The Web APIs support standard VO protocols where possible. They will also support extensions to the VO protocols and additional custom LSST services that are useful for the Portal

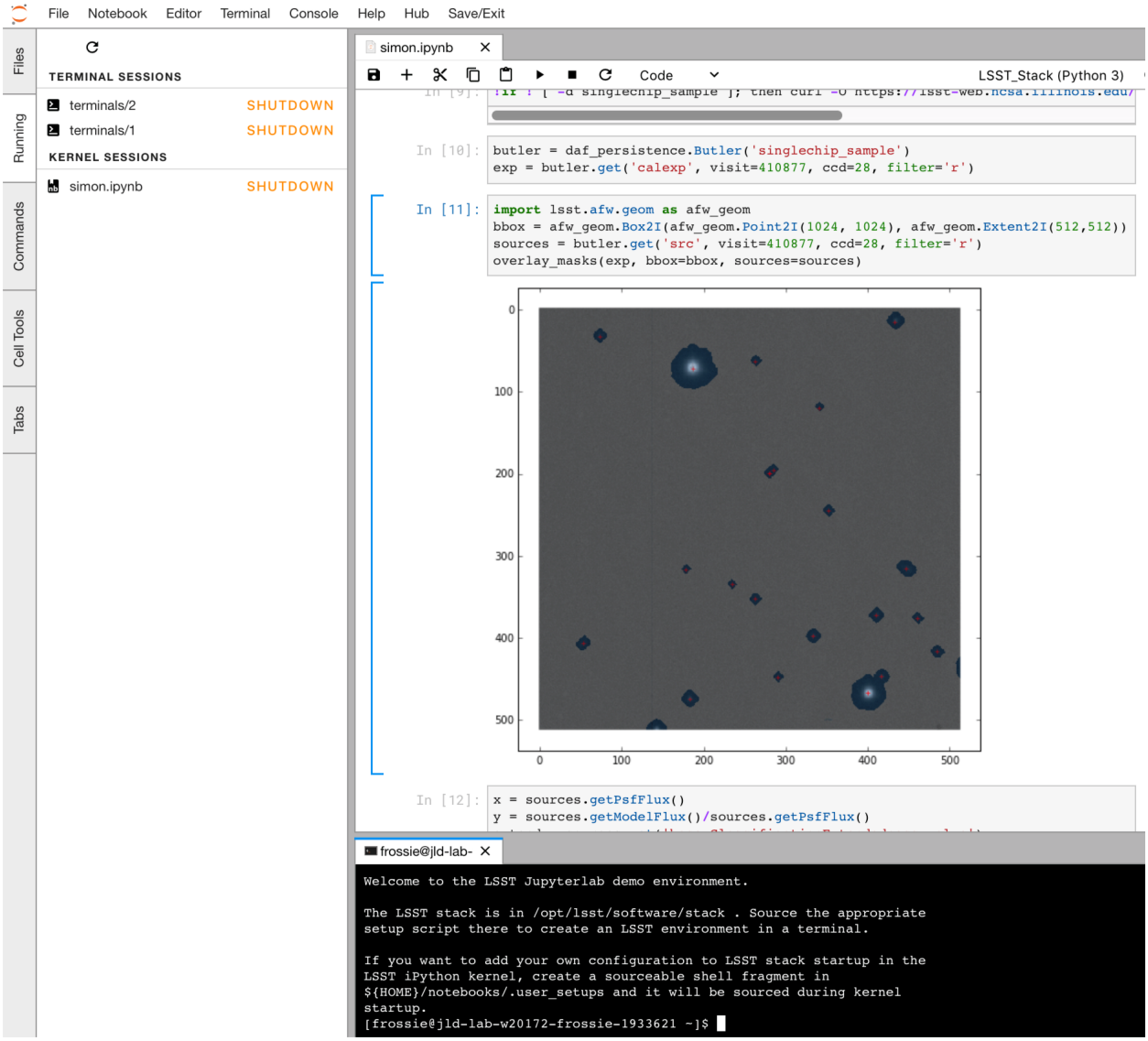

FIGURE 2: Example of star-galaxy separation analysis with the JupyterLab prototype

Aspect or JupyterLab Aspect. To the extent that it makes sense, such extensions will be proposed to the IVOA for incorporation into the standard protocols.

5.2 Catalog and other tabular data access

Catalogs, metadata, provenance, and other tabular data are accessed through the `dbserv` endpoint. For metadata and provenance, table schemas, when possible, will be reused from VO or community standards, such as CAOM2 [1] and others.

5.2.1 TAP service

`dbserv` is the LSST implementation of IVOA's TAP interface. `dbserv` will target compliance with TAP 1.1 once it is recommended by the IVOA.

5.2.2 ADQL implementation

Internal to dbserv is an ADQL parser. This parser will target compliance with ADQL 2.1 once it reaches recommended status. In addition to that, SQL language features of underlying databases will be made available, when possible. Some capabilities will be restricted if a database does not support them, such as subqueries which do not perform well in a distributed database system.

5.2.3 Return data formats

As `dbserv` will be TAP compliant, it will support the VOTable output specified by TAP 1.1, which should be version 1.3. Additional encoding protocols for VOTable output will be supported, including JSON and SQLite output, which were chosen for the ability to represent metadata inline with VOTable data. For all output formats, gzip compression through the HTTP protocol will be supported and encouraged.

5.3 Image data access

Image discovery, metadata, and data retrieval will be available through the `imgserv` endpoint.

5.3.1 Image finding

Image discovery will be facilitated through an implementation of the IVOA SIA V2 interface. Simple implementations of SIA allow direct access to files available via HTTP by adding a link to the SIA response; `imgserv` will use that attribute in the SIA response to link to another service. Image metadata will also be facilitated through TAP, as image metadata will be available according in the database through implementations of the IVOA ObsCore Data Model [**?**] and the CAOM2 data model, as well as LSST-native image metadata representations.

5.3.2 Image retrieval

Images may be retrieved via the URL returned in an SIA request, in the case images can be directly served over HTTP, or via a SODA implementation. The LSST SODA implementation will also offer enhancements to implement project-specific goals without presenting additional interfaces to science users. For example, key/value pairs as used by the Data Butler client may be provided.

5.4 Metadata access

Catalog and image metadata can be accessed via TAP query to provided metadata tables.

5.4.1 Data Releases

Each Data Release has its own set of tables. These are expected to differ in schema between Data Releases. In addition, the image and catalog metadata available for each Data Release will vary. Finally, identifiers within catalogs and images (except raw image identifiers) will vary between Data Releases.

Accordingly, the Data Release number is a key parameter that must be presented to the Web APIs. Collections of datasets that are not part of a Data Release (e.g. in-process, unreleased data or intermediate data products) are assigned labels that are used in place of the Data Release number.

The number of Data Releases available through the LSP and therefore the Web APIs aspect is discussed in section 2.2.8,

5.4.2 Tables

Available tables include those in **?** .

Catalog and image metadata tables will be implemented through IVOA and community standards, when possible, such as those laid out by IVOA's ObsCore DM, VOResource/RegTap, and the CAOM2 data model. Even if direct usage of those models is not feasible, a best-effort export to them will be performed. In such a case, the "native" metadta tables will be documented for direct access by users through TAP.

5.4.3 Table Structure

Each table will have associated metadata that records, for each column, a description, UCD when appropriate, unit when appropriate, and any relationship with other columns (e.g. so several columns making up a covariance matrix can be identified as such without resorting to pattern-matching of column names).

5.5 User Workspace access

WebDAV and/or VOSpace will be provided for user workspace access.

6 The Interconnectedness of the Science Platform

6.1 Sharing Data

6.2 Sharing Queries

6.3 Sharing State

7 Application of the Science Platform inside the Project

7.1 Developer Support

7.2 Commissioning

7.3 Observatory Operations

References

[1] Common Archive Observation Model, URL http://www.opencadc.org/caom2/

[2] Comunidad Federada REUNA, URL http://cofre.reuna.cl/index.php/en/

[3] InCommon: Security, Privacy and Trust for the Research and Education Community, URL https://www.incommon.org/

[4] **[SQR-018]**, Economou, F., 2017, *Investigations into JupyterLab as a basis for the LSST Science Platform*, SQR-018, URL https://sqr-018.lsst.io

[5] **[LSE-163]**, Juric, M., et al., 2017, *LSST Data Products Definition Document*, LSE-163, URL https://ls.st/LSE-163

[6] **[LPM-122]**, Petravick, D., 2015, *LSST Information Classification Policy*, LPM-122, URL https://ls.st/LPM-122