Abstracting the storage and retrieval of image data at the LSST

Tim Jenness¹, James F. Bosch², Pim Schellart², Kian-Tat Lim³, Andrei Salnikov³, and Michelle Gower⁴
¹Large Synoptic Survey Telescope, Tucson, AZ, USA; tjenness@lsst.org
²Princeton University, Princeton, NJ, USA
³SLAC National Accelerator Laboratory, Menlo Park, CA, USA
⁴National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, IL, USA

Abstract. Writing generic data processing pipelines requires that the algorithmic code does not ever have to know about data formats of files, or the locations of those files. At LSST we have a software system known as “the Data Butler,” that abstracts these details from the software developer. Scientists can specify the dataset they want in terms they understand, such as filter, observation identifier, date of observation, and instrument name, and the Butler translates that to one or more files which are read and returned to them as a single Python object. Conversely, once they have created a new dataset they can give it back to the Butler, with a label describing its new status, and the Butler can write it in whatever format it has been configured to use. All configuration is in YAML and supports standard defaults whilst allowing overrides.

1. Introduction

The Large Synoptic Survey Telescope [Ivezic et al. 2008], being built on Cerro Pachón in Chile, will be an automated astronomical survey system that will survey approximately 10,000 deg² of the sky every few nights in six optical bands. The associated Data Management System [Juric et al. 2017; O’Mullane & LSST Data Management Team 2018] is required to process the data from this telescope and publish it as nightly alerts and as annual data releases. The LSST science pipelines (see for example Bosch et al. 2018; Bosch et al. 2019) have been designed such that the algorithmic code is insulated from having to know where data comes from and how it has been serialized. The Butler is the system mediating the storage and retrieval of data, converting Python objects to data files and data files back to Python objects.

2. Butler Components

The Butler consists of a high-level Python API, and three core components: Schema, Registry access, and Datastore. The relationship of these components is shown in Fig. 1. The Schema defines the data model for relating datasets to each other and is defined consistently for all datasets and instruments. The Registry classes allow the
The Butler data model is designed to reflect the relationships between observations and calibrations, and also how the sky can be segmented into different regions, associating each dataset with a particular sky region. This allows you to ask which datasets are needed to calibrate another dataset, which datasets were taken with this filter between these dates, or which datasets would be needed to make a coadd covering this patch of
Abstract data access at LSST

It can also answer provenance queries, such as asking which coadds in a particular filter had at least this number of observations contributing. We are designing the Schema to be generally applicable for astronomical data and we are taking into account that we would like to map our schema to ObsCore (Louys et al. 2017) and CAOM-2 (Dowler 2012) data models in the future.

4. Using the Butler

Individual pipeline tasks work with Python objects. They put datasets and retrieve datasets from the Datastores. The Butler maps a Python object to a serialization format through a “StorageClass” defined in the YAML configuration files for each Datatore. Changing the serialization format from FITS to HDF5 does not require any code changes for the user and is as simple as editing one line in the configuration file. Pre-defined components of a dataset, such as the WCS solution, can be retrieved without reading the full dataset if supported by the formatter. The components supported by each dataset type are defined at the StorageClass level, with code having to be written to assemble a Python composite object from the components and to disassemble a Python object into components.

Below is some user code for retrieving a raw HSC observation along with the relevant flatfield, processing it in some way, and then storing a new version with a different dataset type name. Calibration datasets can be retrieved by knowing the dataset that is to be calibrated.

```python
from lsst.daf.butler import Butler

# Configure a new butler
butler = Butler("config.yaml")

# Specify the requested observation via metadata
dataId = {"instrument": "HSC", "obsid": "HSCA04090000"}

# Retrieve the raw data, process it, and store with new label
raw = butler.get("raw", dataId)
flat = butler.get("flat", dataId)
new = doSomething(raw, flat)
butler.put(new, "newlabel", dataId)

# Get just the WCS without reading the full dataset
wcs = butler.get("newlabel.wcs", dataId)
```

5. Header Translation

To be able to ingest instrument data into a Butler repository, the Butler has to understand some properties of the instrument including filters, detector information, and how to extract metadata from data headers. We have written a separate Python package, astro_metadata_translator, to support header translation and metadata extraction for astronomical instrument headers. The design of this new package has been influenced by the header translator written for ORAC-DR (Jenness & Economou 2015) and unifies the translation systems previously in use at LSST. New translators must be written to allow the Butler to understand data during ingest. Currently, translators exist for
DECam, CFHT MegaPrime, and SuprimeCam and Hyper-SuprimeCam from Subaru, with support for LSST test data being added as needed. This package solely depends on Astropy (Astropy Collaboration 2018) and does not need any LSST infrastructure.

```python
from astropy.io import fits
from astro_metadata_translator import ObservationInfo

hdul = fits.open("hsc.fits")
obsInfo = ObservationInfo(hdul[0].header)
print(f"instrument={obsInfo.instrument}, "
  f"date-obs={obsInfo.datetime_begin}" )
```

6. Summary

The Butler frees you from the worry of file formats and file systems when your main concern is processing and characterizing datasets. The Butler system is not LSST-specific, is written entirely in Python 3 (requiring Python 3.6 or newer following the project baseline (Jenness 2019)), and is driven by external configuration to suit different use cases. The Butler, currently undergoing heavy development and considered to be pre-beta, will be released at the end of 2018 alongside v17.0 of the LSST Science Pipelines. The source code for the Butler can be found at [https://github.com/lsst/daf_butler](https://github.com/lsst/daf_butler), and the source code for the header translator can be found at [https://github.com/lsst/astro_metadata_translator](https://github.com/lsst/astro_metadata_translator).

Acknowledgments. We thank Simon Krughoff, Gregory Dubois-Felsmann, and Meredith Rawls for their reviews of the draft paper. This material is based upon work supported in part by the National Science Foundation through Cooperative Agreement 1258333 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory. Additional LSST funding comes from private donations, grants to universities, and in-kind support from LSSTC Institutional Members.

References

Astropy Collaboration 2018, AJ, 156, 123
Bosch, J., et al. 2018, Publications of the Astronomical Society of Japan, 70, S5
Bosch, J. F., et al. 2019, in ADASS XXVIII, edited by TBD (San Francisco: ASP), vol. in press of ASP Conf. Ser., TBD. [arXiv:1812.03248](https://arxiv.org/abs/1812.03248)
Dowler, P. 2012, in Astronomical Data Analysis Software and Systems XXI, edited by P. Ballester, D. Egret, & N. P. F. Lorente, vol. 461 of ASP Conf. Ser., 339
Ivezić, Ž., et al. 2008, ArXiv e-prints, arXiv:0805.2366
Jenness, T. 2019, in Astronomical Data Analysis Software and Systems XXVII, edited by TBD, vol. TBD of ASP Conf. Ser., TBD. [arXiv:1712.00461](https://arxiv.org/abs/1712.00461)
Jenness, T., & Economou, F. 2015, Astronomy and Computing, 9, 40. [arXiv:1410.7509](https://arxiv.org/abs/1410.7509)
Jurić, M., et al. 2017, in Astronomical Data Analysis Software and Systems XXV, edited by N. P. F. Lorente, K. Shortridge, & R. Wayth, vol. 512 of ASP Conf. Ser., 279
Louys, M., et al. 2017, Observation Data Model Core Components, its Implementation in the Table Access Protocol Version 1.1, Tech. rep.
O’Mullane, W., & LSST Data Management Team 2018, vol. 231 of American Astronomical Society Meeting Abstracts, 362.10