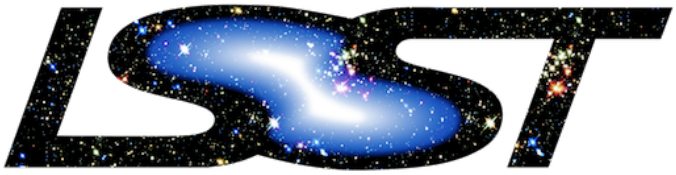

LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)
Data Management

DM Calibration Products

Christopher Z Waters

DMTN-148

Latest Revision: 2020-04-27

Abstract

The goal of this tech note is to provide a plan for the handling of calibration products in the Gen-3 Butler environment, including the construction, validation, certification, and deprecation. This process is equally challenging with the current Gen-2 Butlers, but the hope is that with a clear goal in mind, the currently existing calibrations, for all calibration types and cameras, can be migrated to a common design that will be shared with Gen-3. The current state, the end goal, and the transition steps between these are defined.

Change Record

Version	Date	Description	Owner name
1	YYYY-MM-DD	Unreleased.	Chris Waters

Document source location: `https://github.com/lsst-dm/dmtn-148`

Contents

DM Calibration Products

1 Intro

The goal of this tech note is to provide a plan for the handling of calibration products in the Gen-3 Butler environment. There are a large number of these products currently in use, but many of them do not have a clearly defined process to define, ingest, test, and deprecate them. The lack of documentation on how these processes work has led to many implementations of similar paths, further complicating the issue.

2 The End State

The final state that we wish to get to has the following properties:

- All calibration products are either lsst.afw.image.Exposure image data or an explicit calibration class in the `ip_isr` package.[1]
- All calibration products should be constructed from a DM Task in the `cp_pipe` package, and have the input data and configuration parameters persisted with the calibration itself. This guarantees that any product can be reconstructed in the future. Exceptions will exist for external or lab created calibrations, although the majority of used calibrations will likely be produced directly.
- These provenance, config, and final product files may be saved to an `obs_CAMERA_data` package in git for convenience. This also allows these files to be migrated to other Butler repositories in a consistent fashion. Making these products human readable allows the tracking of value changes with the evolution of the set of exported products.
- All calibration products are accepted for use and have a date range set manually, preventing test runs from being used inadvertently. Calibrations ingested from git should have sufficient metadata that supply appropriate date ranges.
- Calibration information stored in a lsst.afw.cameraGeom object may be used, but is always assumed to be overridden by supplied calibration products. No new calibration

[1] This will require class reorganization to migrate the Defects and Curve calibrations definitions.

information should be stored in these objects, and currently existing calibration information should be deprecated.

3 The Current State

3.1 Standard Calibrations

The classic calibration products, BIAS, DARK, FLAT, and FRINGE, all write FITS files that are indexed in Butler's calibRegistry.sqlite3 repository database. These files are created by pipe_drivers/constructCalibs.p and are ingested into the repository upon acceptance by pipe_tasks/ingestCalibs.py. This ingestion process sets the "validStart" and "validEnd" values to restrict the usage of the calibration, as well as some identifying string which is used (along with calibration type, filter, and detector) to uniquely identify the calibration on disk. We have been using the "calibDate", indicating when the calibration was constructed or ingested, as this string. For calibrations that are generated by cp_pipe, it may be better to use the Butler collection that the calibration was generated in (which will enforce uniqueness on simultaneous processing). Calibrations that are ingested from git could use the repository SHA1 hash, which would provide similar uniqueness.

Although not formally a "classic" calibration product, the SKY calibration is produced, stored, and accessed the same way.

3.2 "User-curated" Calibrations

The other inputs necessary for ISR have generally been described as "user-curated". These products are either defined by the CAMERA as part of the afw.cameraGeom description of the instrument, or are variously persisted and accessed in unique ways. Describing a path to fixing this is a goal of this tech note.

As part of the DEFECTS rework (RFC-595), there is now a fixed and generic way to handle the ingest, if not necessarily the construction, of that product. The QE_CURVE (RFC-625) has been implemented following this process. These DEFECTS-like calibration products have their origins as human-curated text files in the appropriate obs_CAMERA_data packages (and are thereby versioned through the packages' git repositories). The text file must be in a structured format, to allow the header keywords to be read and assigned to a metadata PropertyList.

For these products to be used, they must be converted into a FITS format that can be read by ingestCalibs.py. This is performed by the `pipe_tasks`/ingestCuratedCalibs.py script, which uses the metadata information and the calibration product's python class to read the text version and convert to a FITS file format that can be ingested by ingestCalibs.py.

This updated calibration process removes the necessity that each `obs_CAMERA_data` package implement one-off mapper methods to allow the product to be found. However, only `DEFECTS`, `QE_CURVE`, and `LINEARITY` follow this system, leaving the remaining products to be handled with camera-specific code and cameraGeom entries. In addition to having diverse implementations, these handlers are often rigid, and cannot be versioned easily.

4 Transitioning to Gen3/What We Should Expect From Gen3

4.1 A CalibrationBaseClass

The first step to a consistent calibration system is to ensure that all products in both generations use the `DEFECTS`-like calibration product system. For FITS image products, this requires future calibrations to be constructed with all of the header keyword values listed in table 4.2.

The remaining unconverted calibration types (`CROSSTALK`, `BFKERNEL`) should be converted to use `DEFECTS`-like class, most likely by defining a CalibrationBaseClass, and making all of the calibration products subclasses of that. This would enforce that all calibrations can be relied upon to provide getMetadata, setMetadata, readFits, writeFits, readText, and writeText methods. Gen2 Butler instances would then be able to access all calibration products in a single consistent manner. This also ensures that Gen2 and Gen3 can use the same calibration products.

4.2 Metadata and Header Keywords

The required header keywords are currently poorly standarized. The `OBSTYPE` keyword is universally populated in all calibrations, regardless of if they are generated by constructCalibs.py or are curated by the `obs_CAMERA` packages. Curated calibrations also require the `DETECTOR` and `CALIBDATE`, the last of which is also populated by constructCalibs.py. The `CALIB_ID` is set by constructCalibs.py, and exists for the majority of curated calibrations, but is not consis-

tently used.

The keywords listed in table 4.2 are the proposed list of header keywords that will be required in the new operational calibrations in Gen3. The majority of these are inserted at creation, identifying the content and applicable data range for the calibration. Only upon certification (the replacement for the Gen2 ingest) are the valid date ranges set. To conform with the expected changes to FITS persistence in RFC-686, these and all other calibration-specific metadata fields will be upper-case.

Keyword	Inserted	Description
OBSTYPE	Creation	Calibration dataset type contained in this file.
{OBSTYPE}_SCHEMA	Creation	Storage schema for calibration.
{OBSTYPE}_VERSION	Creation	Storage schema version.
INSTRUME	Creation	Instrument/camera this calibration belongs to.
DETECTOR	Creation	Integer identifier for the ccd/detector.
FILTER	Creation	Name of the physical filter or "NONE".
CALIB_ID	Creation	DataId that can be used to request this calibration after ingest.
CALIB_CREATION_DATE	Creation	Creation date of the calibration.
CALIB_CREATION_TIME	Creation	Creation time of the calibration.
CALIBDATE	Creation	Concatenation of CALIB_CREATION_DATE and CALIB_CREATION_TIME.
VALIDSTART	Certification	Start date the calibration is valid from.
VALIDEND	Certification	End date the calibration is valid from.

4.3 Calibration Construction in Gen3

Calibration products should be constructed using well defined PipelineTask code. This removes the need to delineate between "calibrations" and "user-curated calibrations". The only remaining difference between these two groups is what the initial persisted data type is: "calibrations" are stored in FITS images, as they represent corrections that are only meaningful as full images; "user-curated calibrations" can be visualized in a small number of parameters, and so persisting these in a text format to allow user inspection is reasonable. ECSV and YAML are good formats for these calibrations, and a calibration's readText/writeText methods should implement one of these.

In addition to the calibration product, the PipelineTask code must persist a record of the input data used for construction. This, along with the already persisted pipeline definition and

task configuration, allows the calibration to be regenerated by any user, making the construction process much more transparent than it currently is. A YAML file containing the dataId parameters for all inputs would satisfy this. Appending additional construction statistics (for example, weight factors applied to each input during the combination stage) would be helpful in the validation process.

Newly generated calibration products will belong to a Gen3 collection, and that collection will be used to manage the calibration through the validation and certification process. This collection may only contain one entry for a given Obstype, Instrument, Detector, Filter (OIDF) set.

4.4 Validation

Calibration products are not currently validated, largely due to the opaque construction process. Some set of proper validation tasks must be made, applying the newly constructed product to a set of test exposures prior to that calibration product's certification. Applying these validation tasks continuously as new data is taken will inform the date ranges for which the product is certified. LVV-57 [**?**] lists the requirements that the validation process will need to achieve when implemented.

4.5 Certification

Once a calibration product has been confirmed through validation to correct the instrument effect it is designed for, it will be certified with a date range denoting when the calibration is valid, and assigned to a particular calibration collection (which may contain calibrations the same OIDF values, as the validity range adds another dimension). For FITS image calibrations, this is equivalent to a database operation; no persisted data need change. This is also acceptable for text calibrations, although conversion to FITS (using the CalibrationBaseClass methods) may be desired for IO speed. In addition, as the text calibrations are human readable and easier to manage, they should be exported to the appropriate `obs_CAMERA_data` package. This provides a common versioned repository for the state of the camera calibration that can be exported and used independently of the Gen3 repository where they were constructed. These exports should also include the configuration and input data records.

4.6 Modifying Calibrations

Most calibration products need user intervention only to launch the construction and to validate and certify the result. However, it is assumed that some of the products used (such as `CAMERA` and `QE_CURVE` definitions) will not have a PipelineTask construction process. The version of these products stored in the `obs_CAMERA_data` (or `obs_CAMERA` for the case of the `CAMERA` calibration) repository will be ingested into the Butler repository as needed. These products should still have as much configuration and provenance data as is possible, but should also note any manual edits with a name and date. Enforcing the inclusion of the `obs_CAMERA_data` packages git repository SHA1 entry into the metadata of calibration products that have been ingested from those packages should provide sufficient information to reproduce or revert the manual edits. This provenance method should also be used for calibrations that need to be augmented in non-algorithmic ways, such as `DEFECTS`.

4.7 Deprecating Calibrations

Calibrations that should no longer be used (due to code improvements or newly discovered problems) can be deprecated in Gen3 with operations on the calibration collection that contains them. This will require tools for collection management, but this will be entirely Butler database operations and will likely not be calibration specific.

5 Conclusion

Calibration product	Gen-2 Form	Gen-3 Form	Notes
BIAS			
DARK			
FLAT			
FRINGE			
SKY			
CAMERA			
DEFECTS			
CROSSTALK			
LINEARITY			
BFKERNEL			
QE_CURVE			
STRAYLIGHT			
ILLUMINATION			

A References

References

B Acronyms

Acronym	Description
DM	Data Management
DMTN	DM Technical Note
FITS	Flexible Image Transport System
ISR	Instrument Signal Removal
OIDF	The set of (OBSTYPE, INSTRUME, DETECTOR, FILTER).
RFC	Request For Comment
YAML	Yet Another Markup Language