Rubin Observatory

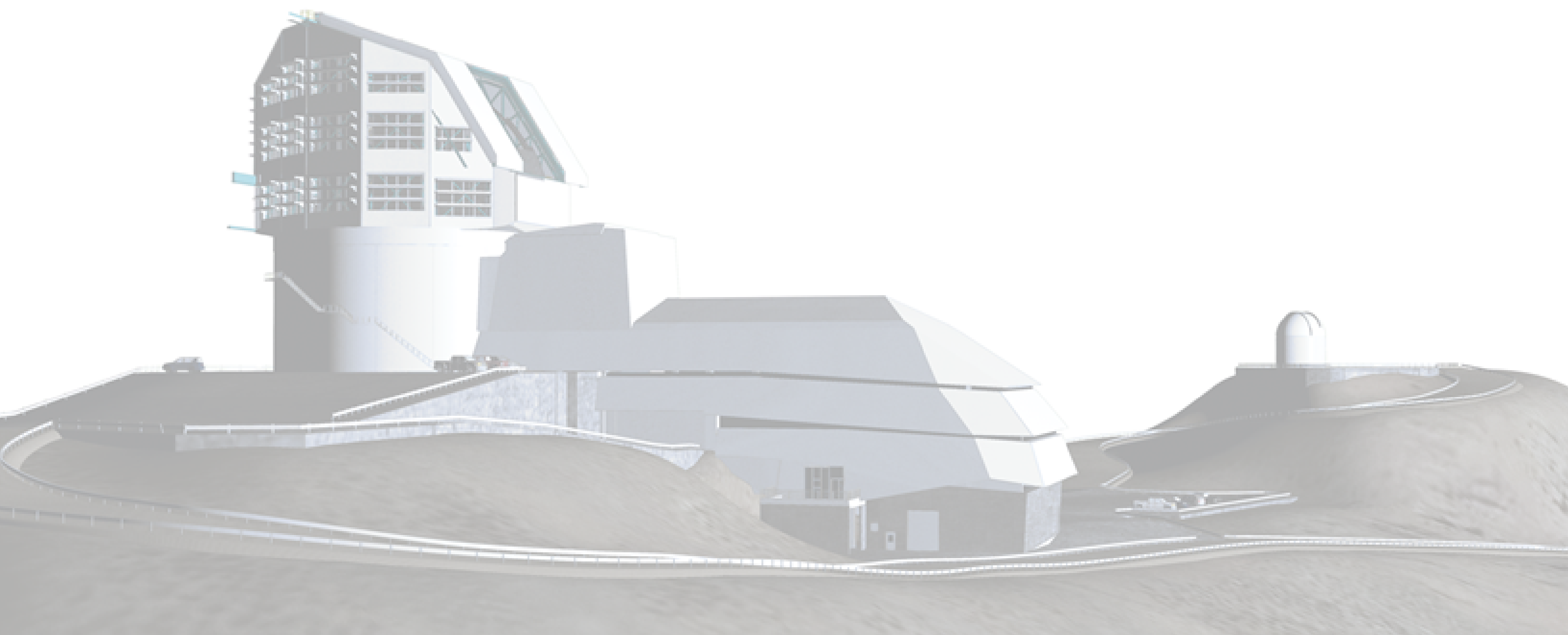
Vera C. Rubin Observatory
Data Management

DM Calibration Products

Christopher Z Waters

DMTN-148

Latest Revision: 2021-01-06



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	2020-04-16	Initial version.	Christopher Waters
2	2020-12-31	Updates to represent how calibrations were actually made.	Christopher Waters

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

Rubin Observatory

Contents

Rubin Observatory

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 a subclass of the `CalibType` calibration class from the `ip_isr` package.
- As much as possible, all calibration products should be constructed by a DM Task in the `cp_pipe` package. The raw input data used for the construction will be available from the Gen-3 Butler, and can be exported along with the task configuration parameters to allow these calibrations to be reconstructed in the future. Exceptions will exist for external or lab created calibrations, although we expect the number of these calibrations to be small.
- The provenance, config, and final product files may be saved to an `obs_CAMERA_data` package in git for convenience. This allows these files to be directly migrated to other Butler repositories without being reconstructed. Making these products human readable allows the tracking of value changes with the evolution of the set of exported products.
- All calibration products will be verified prior to certification. The goal of this process is to demonstrate that the newly constructed calibration satisfies a number of quantifiable quality checks. This should reduce the amount of manual image examination necessary for certification. DMTN-101 will define the exact tests for each calibration type.
- All calibration products are certified for use and have the valid date range set manually, preventing test runs from being used inadvertently. Calibrations ingested from an

obs_CAMERA_data package in git should have sufficient metadata that supply appropriate date ranges.

- The verification pipeline will also be run on new raw calibration data. This will ensure that the certified calibrations being used for the live raw data stream are appropriate, and to monitor the variations of the defined metrics with time.
- 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 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. 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. See the discussion below for more on Gen3 collection naming.

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 are 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 The Gen3 Workflow

Figure 1 illustrates the expected standard workflow. Raw calibration data will be processed through a pipeline defined in the `cp_pipe` package, which will generate a new master calibration. This calibration will be constructed into its own Butler collection, and will not be generally usable. The calibration will be processed through a validation pipeline (possibly depending on additional raw data), producing a set of metrics that illustrate that the calibration satisfies its purpose. Additional processing with the proposed calibration may then be performed to satisfy to a designated "expert" that the calibration should be accepted. The expert then certifies the validity range the calibration is to be used for, and assigns it to a collection of recommended calibrations.

It may be useful to export the calibration products that are processing intensive to the appropriate obs_*_data package. This export process will retain the certification dates, allowing these products to be directly imported into other Butler instances, skipping the intensive processing phase, as illustrated by 2. This procedure can also be used for calibrations that the DM team is not planning to be able to generate, such as detailed quantum efficiency curves. Adding these calibrations to the obs_*_data package will allow them to be formatted appropriately with the relevant metadata, including the validity range, expected for import. These external calibration sources are discussed further below as “curated calibrations”.

Figure 3 illustrates how this process can also be used for manual editing of the calibration. An example of this is the addition of entries to a DEFECTS calibration during the inspection as part of certification. The final edited version of this calibration will be added to the obs_*_data package, along with the metadata and validity range for this calibration, and that can then be re-imported to the Butler and certified for use.

This workflow will also use the Butler collection naming conventions described in DMTN-167. JIRA tickets will be used to coordinate the production of new calibrations, and so those ticket numbers will be used in the production collection, with <instrument>/calib/<ticket>/* containing the production runs for the calibrations, and <instrument>/calib/<ticket> being the calibration collection those calibrations are certified to. These calibration collections will then be connected into a Butler “chained” collection named <instrument>/calib that contains the recommended calibrations for that instrument.

4.2 The IsrCalib Class

The first step to a consistent calibration system is to ensure that all products more complex than a simple image use the new IsrCalib base class. This includes all calibrations in both Gen-2 and Gen-3, to ensure that we can use the same calibration data.

The default storage format for calibration products will be FITS, as that takes advantage of the existing Butler formatter, and matches the expectation for the image based calibrations. A text format should exist for as many calibration types as possible, to allow for a human-readable form to be stored as a curated calibration in the obs_*_data packages.

The IsrCalib is an abstract base class that can be subclassed to implement methods that han-

FIGURE 1: Standard calibration creation.

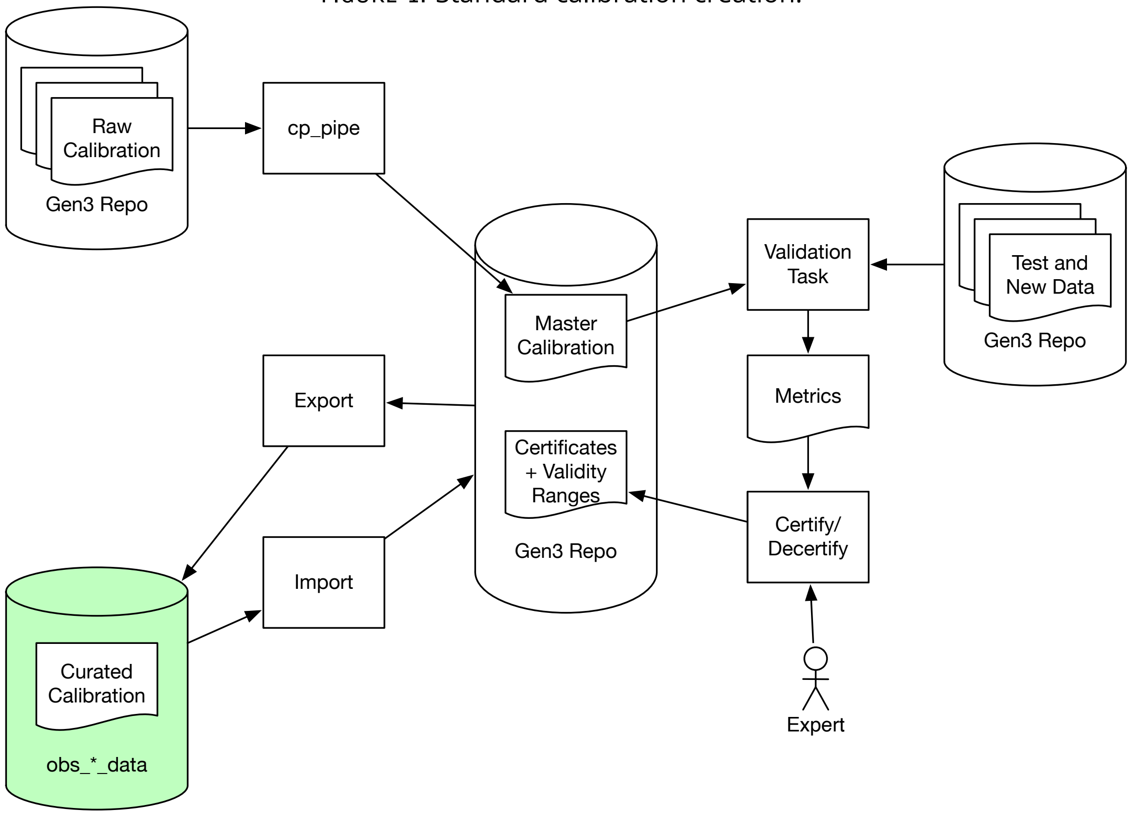


FIGURE 2: External calibration ingest.

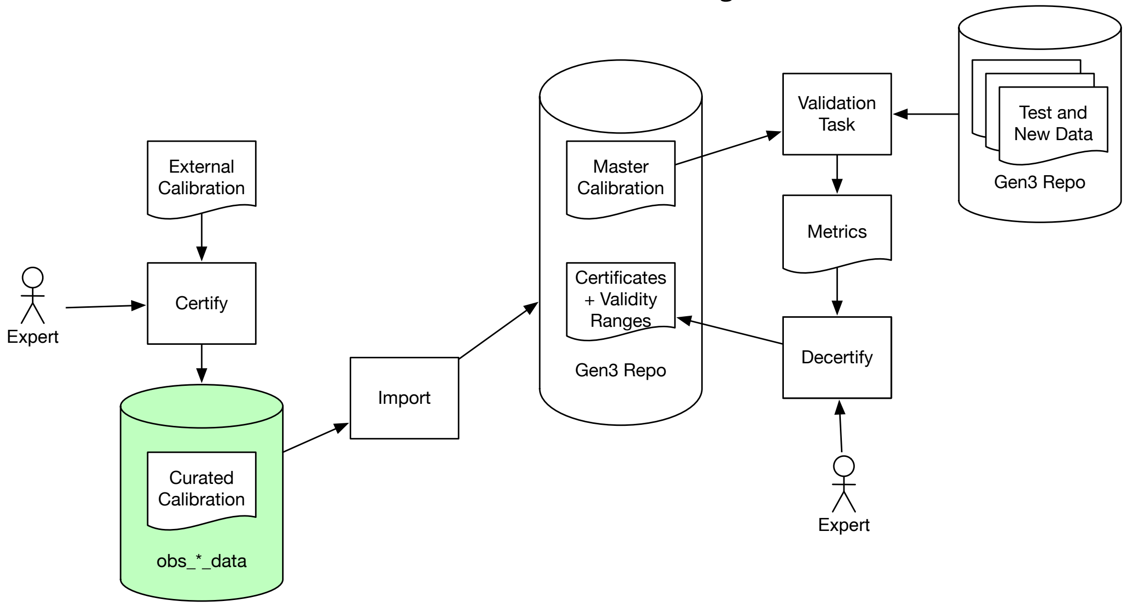
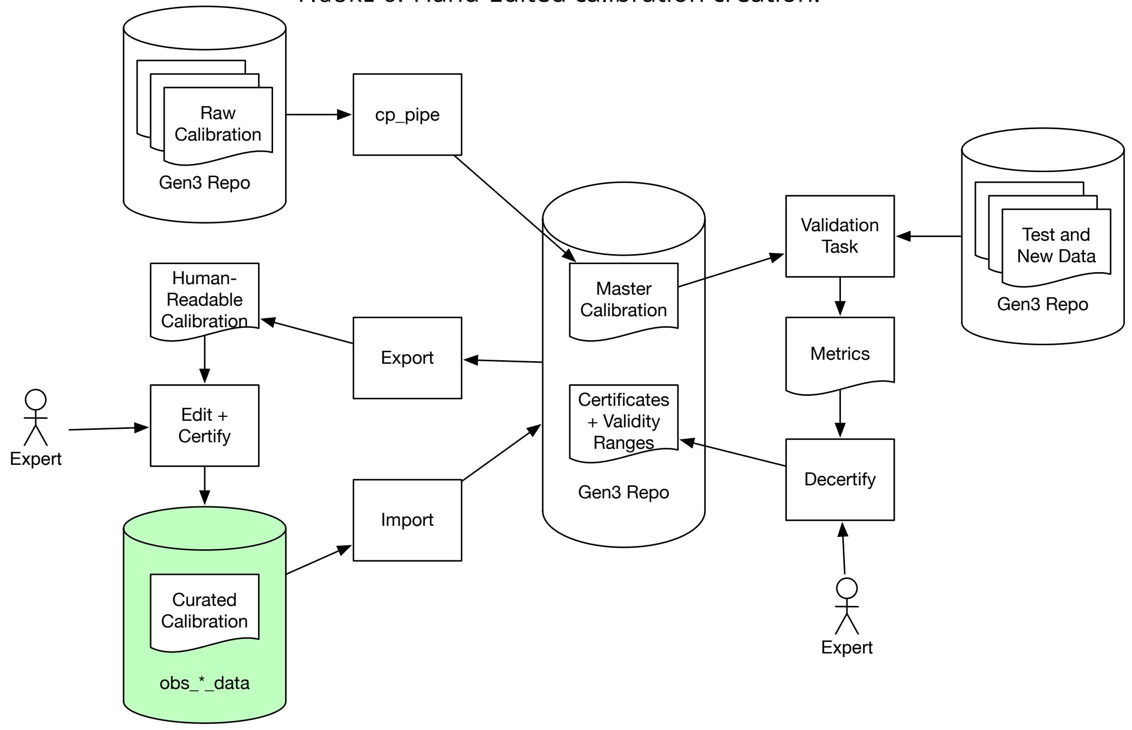


FIGURE 3: Hand-Edited calibration creation.



dle the storage and application of the calibation, as listed in Table 4.2.

Method	Type	Descriptic
fromDetector	Optional	Used to c
fromDict	Required	Construct
toDict	Required	Return a c
fromTable	Required	Construct
toTable	Required	Return a l
validate	Optional	Validate t
apply	Optional	Apply the

4.3 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_*_data 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 consistently used.

The keywords listed in table 4.3 are the proposed list of header keywords that will be required in calibrations in Gen3. 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	Description
	Keywords available as calibration properties.
`OBSTYPE`	Calibration dataset type contained in this file.
`{OBSTYPE}_VERSION`	Storage schema version, to allow backwards compatibility.
`INSTRUME`	Instrument/camera this calibration belongs to.
`RAFTNAME`	Raft containing the detector.
`SLOTNAME`	Slot in the raft containing the detector.
`DETECTOR`	Integer identifier for the detector.
`DET_NAME`	String identifier for the detector.
	Keywords set at creation time.
`CALIB_CREATION_DATE`	Creation date of the calibration.
`CALIB_CREATION_TIME`	Creation time of the calibration.
	Keywords set at creation time from all used inputs.
`CPP_INPUT_N`	Input exposure number.
`CPP_INPUT_DATE_N`	Input observation date.
`CPP_INPUT_EXPT_N`	Input exposure time.
`CPP_INPUT_SCALE_N`	Input scale factor used in combination.
	Keywords that are set on export from Butler.
`CALIBDATE`	As used, is the validity start date for the calibration.
`VALIDSTART`	The start date of the validity range for the calibration.
`VALIDEND`	The end date of the validity range for the calibration.

4.4 Calibration Construction in Gen3

Calibration products are constructed using PipelineTask code defined in the `cp_pipe` package. This package also defines the pipeline definition that connect the new PipelineTasks with those already existing (largely those in `ip_isr`). The goal of this is to minimize the distinction between "calibrations" and "user-curated calibrations" as much as possible, by generating the majority of the calibration products directly from raw images.

The Gen-3 Butler retains a record of all inputs used in the construction of a calibration, and this record will be exported along with the calibration if it is added t 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 OIDF set.

4.5 Validation

Calibration products are not currently validated, largely due to the opaque construction process. Some set of proper validation tasks will be defined, applying the newly constructed product to a set of test exposures prior to that calibration product's certification. It is expected that these test exposures will contain the inputs used in the calibration's construction, as well as similar exposures held back from the construction. Applying these validation tasks continuously as new raw calibration frames are taken will inform the date ranges for which the product is certified. DMTN-101 will discuss this validation further.

4.6 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 `CALIB_ID` values, as the validity range adds another dimension). For calibrations created by PipelineTasks, this is equivalent to a database operation; no persisted data will change.

For calibrations ingested from an `obs_*_data` package, the validity range must be included in the package. Along with the provenance information discussed above, the calibrations These exports should also include the configuration and input data records.

4.7 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`.

4.8 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.

Rubin Observatory

5 Conclusion

Calibration product	Gen-2 Form	Gen-3 Form	Notes
BIAS			
DARK			
FLAT			
FRINGE			
SKY			
CAMERA			
DEFECTS			
CROSSTALK			
PTC			
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