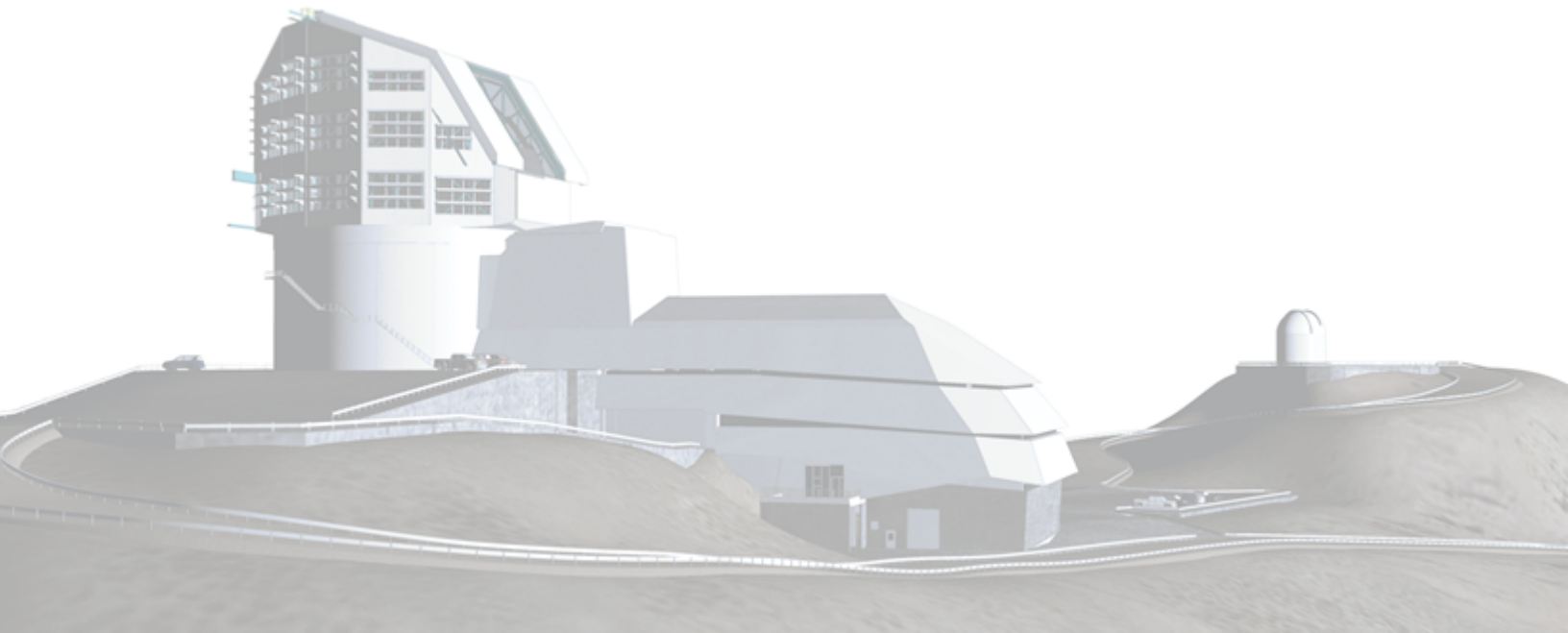

Vera C. Rubin Observatory
Data Management

# Calibration Generation, Verification, Acceptance, and Certification.

**Chris Waters, Eli Rykoff**

**DMTN-222**

**Latest Revision: 2025-04-29**

## Abstract

This technote defines the best practices to be used for calibration generation, the verification that that calibration meets requirements, and when deciding if the calibration should be accepted for use in processing at both the summit and USDF. Other aspects of calibration product management, including the contents of the appropriate “defaults” collection, are also discussed.

# Change Record

| Version | Date | Description | Owner name |
|---|---|---|---|
| 1 | 2022-03-21 | Initial draft. | Chris Waters |
| 2 | 2022-09-16 | Corrected and clarified draft. | Chris Waters |
| 3 | 2024-04-17 | Updated draft reflecting processes actually in use. | Chris Waters |
| 4 | 2024-12-16 | Updated draft with further details on processes | Chris Waters |
| 5 | 2025-02-12 | Updated draft with bps templates and worked example | Eli Rykoff |
| 6 | 2025-04-29 | Updated draft with rucio usage for calibration distribution | Eli Rykoff |

*Document source location:* `https://github.com/lsst-dm/dmtn-222`

# Contents

# Calibration Generation, Verification, Acceptance, and Certification.

## 1 Introduction

The purpose of this technote is to provide guidance on the procedures that are used for the construction and management of calibrations. These guidelines shall be followed for any calibration that will be added to the main butler repositories. For the purposes of this document, we will consider three cases of calibrations.

- Calibrations generated for widespread use, using the main butler repository. These are referred to as "combined calibrations" below, and indicate the calibrations that are used for science processing.
- Curated calibrations that are defined by an `obs_` package and must be ingested to the butler repository, as they cannot be directly generated from raw data. The camera geometry calibration is an example of this type of calibration.
- Calibrations that have been exported from one butler repository for use in another.

Additional private calibrations produced for tests may also exist, but as those will only exist in a user-space collections, they will not be discussed further in this document.

Briefly, calibration construction involves the following steps:

**Generation** An appropriate set of exposures is chosen and processed through the correct `cp_pipe` pipeline.

**Verification** The proposed calibration is used to process exposures through the matching `cp_verify` pipeline. This processing measures a set of quality metrics, as defined by DMTN-101, to determine if the newly made calibration meets requirements.

**Certification** The proposed calibration is certified for a particular usage date range. These are generally open ended, with only the start date defined. We expect to know the start date for the majority of all calibrations, as they should correspond with changes to the camera.

**Approval** The TAXICAB considers the proposed calibrations and their associated verification results, and makes the decision on whether the proposal is accepted for use.

**Distribution** The collection containing the new calibrations are included in the main calibration collection chain, for all repositories that need the updated calibration.

Figure 1 displays the relationship between the various stages of construction, validation, and use of combined calibrations. **CZW:** Is this still useful?

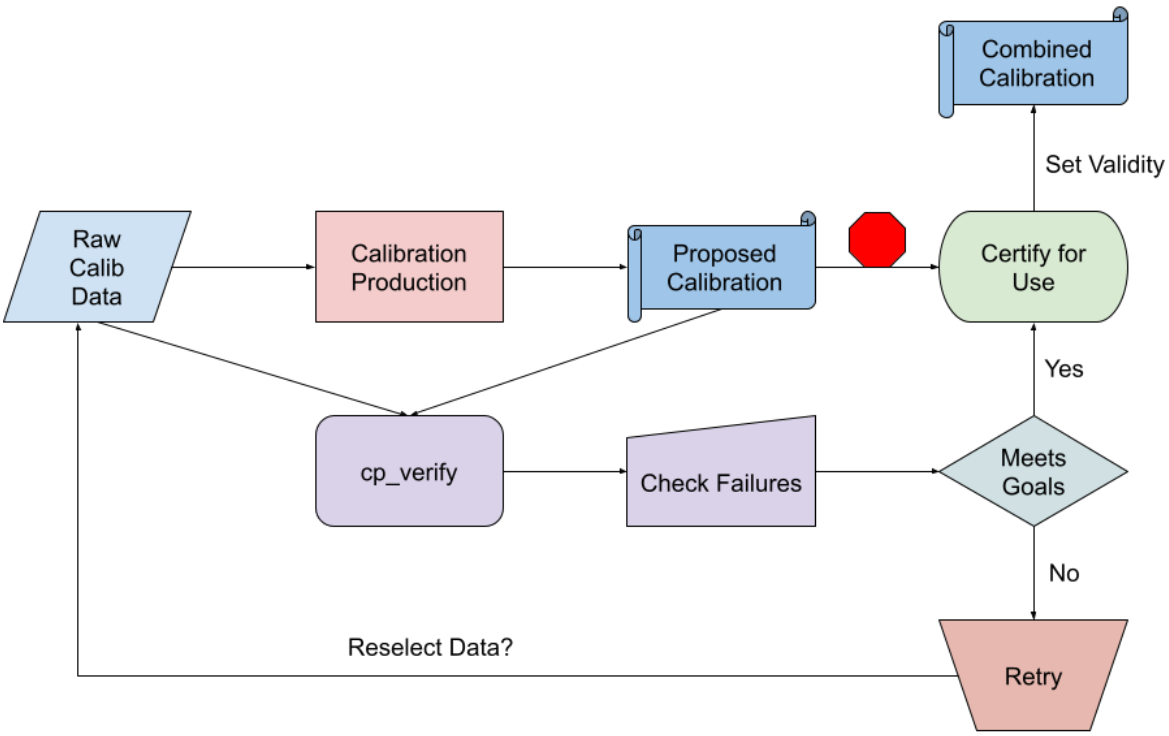


FIGURE 1: Flowchart of the calibration construction process.

## 2 Collection naming

Consistent collection names make the management of calibrations easier. JIRA tickets are used to ensure that these collection names are unique, and that there is a clear location to find

the construction artifacts for later analysis. In addition to this ticket, a short string explaining the purpose of the calibration set should be included in the collection name to provide a human readable "tag." The following collection name patterns, based on the recommendations in DMTN-167 should be followed for all calibrations that will be approved by the TAXICAB.

The calibration generation should use the form

```
$INSTRUMENT/calib/$TICKET/$TAG/${CALIB_TYPE}Gen.${RERUN_ITERATION}
```

where `$INSTRUMENT` is the camera name, `$TICKET` is the JIRA ticket value, `$TAG` is the short human readable string, `$CALIB_TYPE` is the calibration type being generated, and `$RERUN_ITERATION` is a date string of the form `YYYYMMDDv` indicating when the calibration was made, with a trailing character to be incremented if the generation must be retried. As an example, a hypothetical new bias would have a collection name like `LATISS/calib/DM-12345/voltageChange/biasGen.20220915a`.

For verification, a similar form is used:

```
$INSTRUMENT/calib/$TICKET/$TAG/verify${CALIB_TYPE}.${RERUN_ITERATION}
```

with the same elements as for generation. The certification process (see below) also creates a new CALIBRATION collection, which should just contain the calibration type:

```
$INSTRUMENT/calib/$TICKET/$TAG/${CALIB_TYPE}.${CERTIFICATION_RERUN}
```

These CALIBRATION collections should be added to a CHAINED collection that uses the collection base only up to the ticket. This ticket-level CHAINED collection provides a way to add and remove all of the calibrations constructed as part of that ticket in one operation. In addition, by ensuring that new calibration collections are prepended to the top level CHAINED collection, we can avoid needing to set end dates during calibration certification. The butler search for calibrations completes when the first matching calibration is found. So, as an example, by putting a collection with a start date of 2024-10-01 before one with a start date of 2024-06-01, we can ensure that an exposure taken on 2024-10-15 will be processed with that newer calibration, but one taken on 2024-09-01 will be processed with the older one.

# 3 New Combined Calibrations Construction

A record of the calibration construction process should be retained and attached to the JIRA ticket managing the work, with all commands executed and exposure selections recorded. Having this record will allow for understanding what happened during construction, in case the final products have problems.

## 3.1 Generation

Combined calibrations will be generated directly from raw exposures as much as possible. The tasks and pipelines in the `cp_pipe` package can produce all of the calibrations that are currently used for image processing, and can be supplemented as new corrections are developed. The main documentation for calibration construction is included in `cp_pipe` at `https://pipelines.lsst.io/v/daily/modules/lsst.cp.pipe/constructing-calibrations.html`, but the main points will be summarized here.

Calibrations are inter-dependent, and so the construction of one type may require precursor calibrations to be built first. Figure 2 shows the current dependence, with each box pointing to the calibrations that they depend on. The result of this is that changes in one calibration (such as the gains derived from the photon transfer curve) require other calibrations (the linearity, the brighter-fatter kernel, and the charge transfer inefficiency) to be built as well.

The `observation_type` and `observation_reason` of the input exposures should match the calibration type to be constructed, with the exception of the fringe and crosstalk calibrations, which are constructed from science exposures. Most calibrations can be constructed from a single set of daily calibrations, with the number of bias, dark, and flat frames in these sets (generally of order 15-20) sufficient to create a usable combined calibration. Dense PTC curves will require many more inputs (on the order of 100 pairs of exposures), and we currently expect that we will have dedicated observation sequences for this purpose.

Calibrations constructed for general use should be able to use the version of the `cp_pipe` tasks and pipelines on the main github branch. It is preferable to keep code development separate from the calibration construction, but it is expected that these will likely be coupled during commissioning.

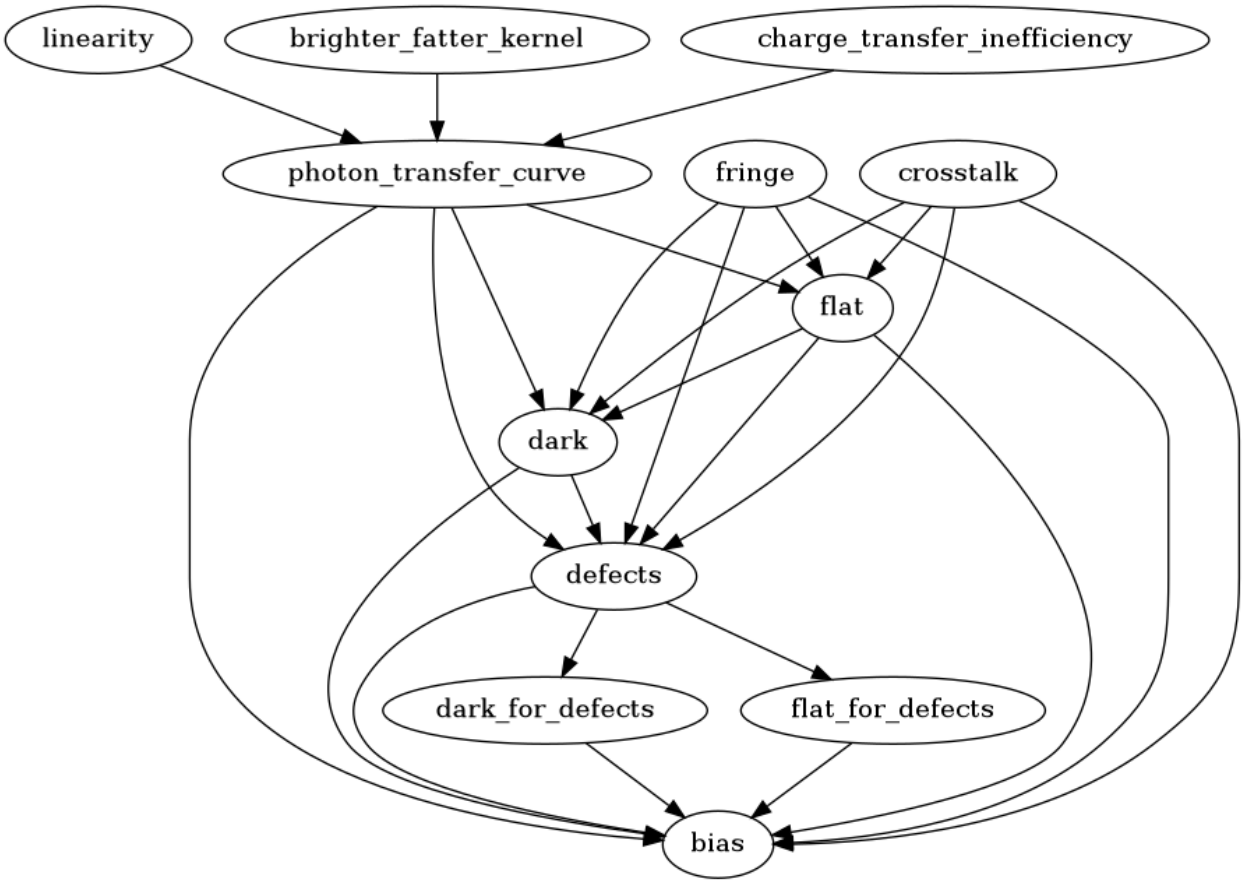


FIGURE 2: Dependency charge of calibration products. The arrow indicates the parent calibration.

To ensure all butler repositories have a consistent set of calibrations, we have decided that only one processing location should perform the calibraion construction steps. The US Data Facility (USDF) is now operational, all calibrations used for the survey will be generated there. The process for transferring the calibrations to other butler sites is discussed below in Section 3.5.

## 3.2 Verification

A verification report is then generated from the generation and verification collections, and will be supplied to the Telescope And auXiliary Instrumentation Calibration Acceptance Board (TAXICAB), as defined in `https://rtn-075.lsst.io/`.

Once the proposed calibrations have been generated, the calibration should be used for processing using the `cp_verify` tasks and pipelines. These tasks measure quality metrics from those processed exposures, and identify any test failures. At a minimum, the exposures used to construct the calibration should be included, as this can identify problematic inputs that degrade the calibration quality. An example of this is saturated flat exposures, which do not flat-field well, and should not be included in the final flat calibration. In running the `cp_verify` tasks, the input butler collections specified should have the construction RUN collection placed at the beginning of the list, to ensure that the verification process will find and use the calibration we wish to verify.

Exposures from outside the set used for construction should be added to provide insight into the expected validity range for the calibration. For calibrations generated to correct for known camera changes (such as a sequencer upgrade), we will likely need to generate the combined calibration from a limited set of inputs, and will therefore rely on `cp_verify` metrics to monitor those calibrations. In cases where we have multiple days of observations that can be used, including exposures from multiple dates in the verification set can be used to As long as the metrics on those exposures remain within the limits defined in DMTN-101, the calibration may continue to be valid for the date range including those additional exposures. This can be used to establish the valid date ranges to be used when certifying the calibration.

The `cp_verify` pipelines will generate and publish `analysis_tools` “core” metrics and plots to cover the DMTN-101 tests. These metrics and plots will also include useful diagnostic results based on the camera team `eo_pipe` tests. Further “extended” metrics and plots may also need to be generated to supply additional debugging information about the calibrations.

The `cp_verify` report is generated by a command such as

```
$CP_VERIFY_DIR/bin/cpv_report.py -r /repo/embargo \
                    -O ~/public_html/cpv_reports/TAXICAB-15 \
                    -c LSSTComCam/calib/DM-47447/gainFixup/verifyFlat-g.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/verifyFlat-r.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/verifyFlat-i.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/verifyBias.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/verifyDark.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/flatGen-g.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/flatGen-i.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/flatGen-r.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/darkGen.20241107a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/atoolsDark.20241108a \
                    -c LSSTComCam/calib/DM-47447/gainFixup/atoolsDarkDet.20241108a
```

As many collections and types of calibrations can be added to the list of inputs. All applicable verify collections should be included, and although the generation collections should be linked from the verify collection (allowing them to be searched for data products), it is generally safe to add them to the list as well. This report constructs set of HTML documents and web-friendly images so the calibration quality can be checked. The report is known to be incomplete, as there is a large backlog of tickets for adding plots and metrics to the code.

## 3.3 Certification

Once the new combined calibration has been generated and verified, it can be certified for use for a given date range. Calibrations that have been constructed due to a camera or telescope change, or that are being built to replace another calibration that is no longer within the test specifications, should always have a starting validity date, with the end date left open. This ensures that future data taken will always have valid calibrations for processing.

If historical calibrations are being constructed, the end date should be known from the daily calibration processing results stored in the visit database (see below). Future development is needed to allow calibrations to be recertified to update the date ranges.

The certification process can be done with a simple butler command:

```
butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    $INSTRUMENT/calib/$TICKET/$TAG/biasGen.$RERUN \
    $INSTRUMENT/calib/$TICKET/$TAG/bias.${CERT_RERUN} \
    bias
```

where the `$START_DATE` is the ISO-8601 datetime in TAI coordinates, and `$CERT_RERUN` is a dated rerun string as used above (as the generation and certification may take place on different dates).

Note that certification is purely a "database operation," with the certified collection gaining a "link" to the dataset in the construction collection. This means that if the underlying generation collection is removed, the calibrations may be lost as well.

## 3.4 Approval

With the calibrations built, verified, and certified, a TAXICAB "hailing" ticket should be created, with a verification report attached for consideration. Any member of the Rubin Observatory Team is welcome to join the TAXICAB meeting, and the meeting will make decisions as a consensus. Any additional processing that is suggested by the TAXICAB should be defined and run prior to the TAXICAB meeting, which will have a planned weekly timeslot. Currently, this meeting is scheduled for Tuesday at 2:30pm Project Time. If no open TAXICAB hailing tickets exist, this meeting will be skipped. Scheduling a TAXICAB meeting generally indicates that the associated ticket has been marked as "Flagged," as it is awaiting the decision on acceptance.

The TAXICAB will consider the verification reports, identify any potential issues with the calibration set, and determine if any verification test failures warrant restarting the construction process to address the issues. Ideally, all verification metrics will succeed, and a quick check of residual exposures will show no unexpected features. In the more likely case that some fraction of these tests fail, the TAXICAB will be tasked with deciding if the failures are fatal and that the calibration should be fully rejected, or if the failures are small enough in number or impact that the calibration can be accepted for use despite them. The TAXICAB will operate on a consensus basis, to ensure that all stakeholders have input on this process.

If the calibrations were built using a ticket/development branch of any software, those code changes must be reviewed and approved through the standard DM process prior to hailing the TAXICAB. If no new code was added, then the approval of the TAXICAB can be used as the review process to close the initial generation ticket.

Managing the approval for calibrations will follow a three-ticket process. As described above, a construction ticket is used to manage all construction, verification, and certification steps, and is included in all output collection names. A TAXICAB ticket is used to manage the approval of the calibrations, and to contain pointers to the verification report, the generation ticket, and the final ticket, which manages the deployment of these calibrations. This deployment ticket is used for copying the calibration into all necessary public repositories, as described below. If code was added as part of the construction, that work should be reviewed on the construction ticket prior to the TAXICAB meeting. Otherwise, the TAXICAB approval (by marking that ticket "Adopted") should also mark the construction ticket as reviewed. The deployment ticket will be self-reviewing in the future: a trivial processing task should be run at each repository that has had the new calibrations deployed, with checks that those new calibrations are correctly selected and used.

A summary of the TAXICAB tickets is currently compiled on the project Confluence `https://rubinobs.atlassian.net/wiki/spaces/DM/pages/281673762`. This page should be updated as new TAXICAB tickets are generated, to ensure that we have a central location to track the various tickets. As some repositories may have the distribution step skipped for various reasons, such as not having data for the camera the calibrations were created for, this page also lists which repositories the calibrations on that ticket were copied to. A central archive of the exported calibrations needs to be established, so that the calibration CHAINED collection can be reconstructed at all data facilities.

## 3.5 Distribution

Upon approval of the TAXICAB, the calibrations can be distributed for use. A separate distribution ticket should be created to handle this work, and linked to both the construction ticket and the TAXICAB ticket.

There are three different distributions that need to be handled. First, calibrations in `repomain` need to be chained into the default calibration collection. Second, calibrations generated in `repomain` need to be copied over to the `embargo` repo at USDF if they need to be applied to

embargoed images. Third, calibrations need to be copied to the Summit, Base Test Stand, and other Data Facilities via `rucio`. In the latter two cases, after calibrations have been copied they need to be chained into the default calibration collection.

In most cases, when new forward-looking calibrations are deployed as the default, the new CHAINED collection which was created as part of the calibration generation can be prepended to the top level calibration CHAINED collection, installing the calibrations for use by default.

### 3.5.1 Exporting and Importing Calibrations

The calibrations must then be exported for use in other repositories and to create a local archive of calibrations at USDF. An example command to do such is

```
butler export-calibs $REPO \
       ./export_directory \
       LATISS/calib/DM-XYZ \
       LATISS/calib/DM-XYZ/voltageChange/bias [...]
```

This command exports the files into the `export_directory` location, and constructs a YAML description of the calibrations and their collections. Exporting the ticket-level CHAINED collection will export all of the children CALIBRATION collections, making it the preferred way to export calibrations.

This `export_directory` must then be transferred to the location of the new repository, where it can be imported with the command

```
butler import $NEW_REPO  --transfer copy \
       --export-file ./export_directory/export.yaml ./export_directory \
       -s instrument -s detector -s physical_filter
```

The `--transfer copy` is strongly suggested, as this will copy the files into the repository datastore, removing any dependency on the `export_directory`. The three `-s` arguments indicate that the `instrument`, `detector`, and `physical_filter` definitions contained the the YAML description should be skipped, as they will already exist in a repository that has been set up for the appropriate camera.

The newly imported collections will not by default be part of the public calibration collection. To do so, the new collections must be added to the collection chain. Using the following command with the 'prepend' mode will add the new collections to the start of the collection chain, making them available.

```
butler collection-chain $NEW_REPO --mode=prepend LATISS/calib \
       LATISS/calib/DM-XYZ \
       LATISS/calib/DM-ABC
```

The distribution ticket should be able to be self-reviewed, after confirming that at least one exposure from the validity range of the new calibrations can be processed through `IsrTask`, and that the output processed exposure has the correct calibration information recorded in its header.

The following table lists the current set of facilities, repositories, and which cameras are deployed in that repository.

| Data Facility | Repository | Camera |
|---|---|---|
| USDF | embargo_old | LATISS |
| | | LSSTComCam |
| | | LSSTComCamSim |
| | | LSSTCam |
| USDF | /repo/embargo | LATISS |
| | | LSSTComCam |
| | | LSSTComCamSim |
| | | LSSTCam |
| USDF | /repo/main | LATISS |
| | | LSSTComCam |
| | | LSSTComCamSim |
| | | LSSTCam |
| USDF | /repo/ir2 | LSSTCam |
| Summit | /repo/LATISS | LATISS |
| Summit | /repo/LSSTComCam | LSSTComCam |
| | | LSSTComCamSim |
| Summit | /repo/LSSTCam | LSSTCam |
| Tucson Test Stand (TTS) | /repo/LATISS | LATISS |
| Tucson Test Stand (TTS) | /repo/LSSTComCam | LSSTComCam |
| | | LSSTComCamSim |

### 3.5.2 Distribution With Rucio

The way to copy calibration files from `repomain` to the Summit, BTS, and the other DFs is via `rucio`. Note that `rucio` does not have access to the `embargo` repo.

Using `rucio` involves first registering the datasets with the `rucio` service, and then creating a "rule" which says where the files should be synchronized. Once the rule is created then the `rucio` service will do the necessary synchronizations, and will place the calibrations in special S3 calibration buckets at BTS, the Summit, and at any other DFs that are configured. The S3 calibration buckets are configured such that only `rucio` has write access, and these files cannot be directly deleted (on purpose or accident). We have not yet figured out the plans for removing old calibrations.

At the current time `rucio` will only synchronize the files, and it is the responsibility of the person

doing the deployment to copy over the above `export.yaml` and do the ingestion and the final collection chaining. In the near future we expect the ingestion will be done automatically, though explicit deployment chaining is going to be a manual task for the foreseeable future.

The following is how a set of flats can be synchronized. Note that the `DSLABEL` can be anything. But using the ticket name is very important to ensure that the dataset name is unique for each set of registered calibrations.

```
# Define a descriptive label
DSLABEL=flat

# This is the collection where calibs will be copied from
COLLECTION=$INSTRUMENT/calib/$TICKET
# This dataset name must be unique
DATASET=Dataset/$INSTRUMENT/$DSLABEL/$TICKET

# Loop over all the types of calibration products in the
# collection. In the future rucio-register may be able
# take an array of dataset types.
# Be careful with this step, as you need to know all the
# types of calibs in the collection.
for dstype in flat; do
  rucio-register data-products \
      -s 10 \
      -C /sdf/data/rubin/shared/calibration_archive/rucio/main-calib-config.yaml \
      -r /repo/main \
      -t $dstype \
      -c $COLLECTION \
      -d $DATASET
done

# Close the dataset so that no further files can be
# associated with it.
rucio did update --close --did ancillary:$DATASET

# List the contents of the dataset to confirm that everything
```

```
# looks as expected.
rucio did content list --did ancillary:$DATASET

# Define a rule to synchronize the files at 3 sites (note that
# the number of copies must match the number of rucio sites).
RULEID=$(rucio rule add \
    --rses 'SLAC_BUTLER_DISK|BASE_CALIB_DISK|SUMMIT_CALIB_DISK' \
    --did ancillary:$DATASET \
    --copies 3)
# Show detailed information on the synchronization of the files.
rucio rule show --rule $RULEID

# Show a concise table of the synchronization state of all the
# files and locations. This command should be run every once in
# a while until all the files have completed synchronization.
rucio replica list dataset --did ancillary:$DATASET
```

Once the replica list shows that everything is synchronized among the sites, the `export.yaml` can be copied to the BTS and Summit and imported as such. It is very important to get the file prefix correct, as well as to use the `direct` import method which tells the database that the files are already safely in the calibrations bucket.

```
butler import $REPO s3://butler@rubinobs-calibrations/rucio/ancillary --export-file `pwd`/export.yaml
```

After this the calib collection chain can be updated as with main and embargo.

## 3.6 Auxilliary Data Products

In addition to calibrations that are directly used in the processing of data, there are other data products that are managed and distributed by the calibrations team to ensure that repositories are consistent. These datasets are generally added to the `$INSTRUMENT/defaults` CHAINED collection, which links the raw exposure collection, the top level CHAINED calibration collec-

tion, and other standard data products that are needed for completely processing. The two most common of these data products are reference catalogs (refcats) and skymaps.

### 3.6.1 Refcats

**CZW:** This needs to be written.

### 3.6.2 Skymaps

The skymaps contain the spatial information about the tracts and patches used for coadd construction. The standard skymap configuration files can be found in **CZW:** a TBD archive . The information about these skymaps is stored in the butler database, and not as standard on-disk files, so care is needed to ensure that the database has enough storage space for the 3-5GB used by each skymap.

Registering the skymap is easy when using a predefined configuration:

```
butler register-skymap $REPO \
       -C skymap-lsst_cells_v1.config
```

As these are stored in the database, a butler repository can have the available skymaps checked by running

```
butler query-dimension-records $REPO skymap
```

### 3.6.3 Other Data Products

Currently, FGCM lookup tables and “pretrained models” are the other data products that are connected to the default collection. **CZW:** This needs written as well.

## 3.7 Calibration Construction Checklist

- File DM ticket for calibration construction that will hold the details of that process

- Select inputs for calibration
- Run generation pipeline from `cp_pipe`.
- Run associated verification pipeline from `cp_verify` on the same inputs, supplemented with similar exposures to test for time variability.
- Create verification report.
- File TAXICAB ticket with verification report and any other information.
- File deployment ticket.
- Upon approval, export/import the calibrations to all other repos. This may be further automated in the future.
- Ensure summit calibrations are tagged correctly.

# 4 Daily Calibrations

Daily calibrations are used to monitor the camera and telescope for changes. The daily calibration processing will simply verify these newly taken exposures against the existing calibration set as shown in Figure 3. This allows the long-term stability of the calibrations to be monitored. We do not plan to ever generate new combined calibrations automatically from the daily calibration scripts.

The `cp_verify` pipeline from the daily processing should run using the "butler+sasquatch" butler repos, which ensure that the `analysis_tools` metrics are dispatched to the summit Chronograf (https://summit-lsp.lsst.codes/chronograf). There will be a new dashboard available to display these metrics as they change from day-to-day, providing a way to monitor camera/system changes.

The verification results from the daily calibration processing will evenutally issue LOVE-based alarms if any tests fail. This should notify the calibration team members and result in an investigation to determine if updated calibrations need to be produced.

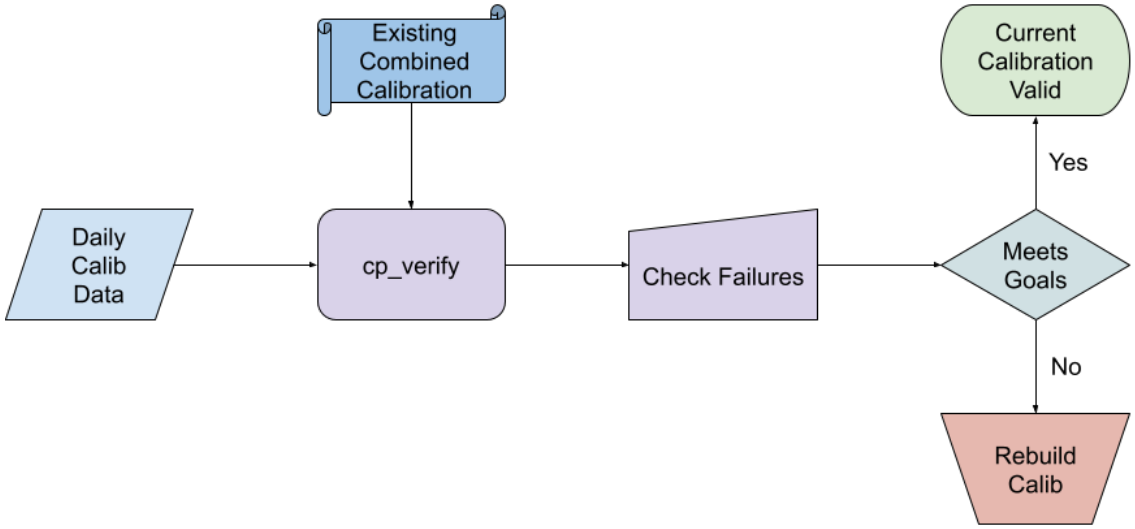


FIGURE 3: Flowchart of the daily calibration process.

# 5 Curated Calibrations

Curated calibrations are those calibrations that cannot easily be generated from a series of exposures, or that require special hardware that will not be available at the summit. Currently, the camera geometry calibration is the most important curated calibration in wide use. These calibrations are ingested via the `butler write-curated-calibrations` command. This command by default will attempt to write to the main `$INSTRUMENT/calib` collection. This is generally not desired, as it is useful for that collection name to point to a CHAINED butler collection, to allow for the calibration management process described above. Instead, a JIRA ticketed collection name should be used, as the following example illustrates for the LATISS camera:

```
butler write-curated-calibrations $REPO lsst.obs.lsst.Latiss \
       --collection LATISS/calib/DM-XYZ --label DM-XYZ
```

This will ensure that the calibrations can be chained into the main collection as detailed above.

# 6 Batch Process Submission (BPS) Templates

We have put together a set of convenient bps[1] template files that have been tested at USDF but should also be suitable testing at other locations. These templates are intended for batch processing of calibration construction and verification for both user collection testing and final calibration production. This section has a brief overview, but please see the README files at `https://github.com/lsst/cp_pipe/tree/main/bps/templates/README.md` and `https://github.com/lsst/cp_ver` for up-to-date information. Note that the templates assume that each calibration is certified into the overall calibration chain called `${USER_CALIB_PREFIX}${INSTRUMENT}calib${TICKET}${TAG}` between steps. A walkthrough example is in Section 7.

Use of the templates are controlled by a number of environment variables. At the writing of this documentation these include:

- `export USER_CALIB_PREFIX=""` or `export USER_CALIB_PREFIX=üerykoff`: Set to an empty string for official calibrations, or the user collection prefix with trailing slash.

[1] https://pipelines.lsst.io/modules/lsst.ctrl.bps/quickstart.html

- `export INSTRUMENT=LSSTComCam`: The name of the instrument.
- `export TICKET=DM-46562`: The name of the ticket associated with the calib construction.
- `export REPO=repomain`: The name of the butler repository to generate calibs.
- `export RAW_COLLECTION=LSSTComCamrawall`: The name of the raw data collection.
- `export CALIB_COLLECTIONS=LSSTComCamcalibDM-46825`: Comma-separated list of curated or previously generated calibration collections to use as a starting point.
- `export TAG=newCalibs`: A human-readable tag to help indicate why a set of calibs were built (matched to the ticket number).
- `export RERUN=20250122a`: The rerun name to indicate when the calibrations were generated.
- `export BOOTSTRAP_RUN_NUMBER=1`: The bootstrap run number ensures that bootstrap run collection names are unique in case of an initial mistake.
- `export SELECTION_BIAS=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to make the bootstrapBias and bias frames.
- `export SELECTION_DARK=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to make the bootstrapDark and dark frames.
- `export SELECTION_FLAT_BOOTSTRAP=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to make the bootstrapFlat.
- `export SELECTION_PTC=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to generate the PTC.
- `export SELECTION_PTC_LINEARIZER=$SELECTION_PTC`: The selection of raws to generate the linearizer; usually will be the same as the PTC selection.
- `export SELECTION_PTC_BFK=$SELECTION_PTC`: The selection of raws to generate the brighter-fatter kernel; usually will be the same as the PTC selection.
- `export SELECTION_PTC_CTI=$SELECTION_PTC`: The selection of raws to generate the charge-transfer-inefficiency dataset; usually will be the same as the PTC selection.
- `export SELECTION_FLAT_g=ïnstrument='LSSTComCam and selection_stringë`: The selection of raws to generate the g-band flat. There will need to be one selection for each ugrizy.

Additionally, for the verify templates we have:

- `export VERIFY_RERUN=20250122a`: The rerun name to indicate when the verification was run.
- `export SELECTION_BIAS_V=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to verify the bias frames.
- `export SELECTION_DARK_V=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to verify the dark frames.
- `export SELECTION_PTC_V=ïnstrument='LSSTComCam' and selection_stringë`: The selection of raws to verify the PTC.
- `export SELECTION_PTC_LINEARIZER_V=$SELECTION_PTC_V`: The selection of raws to verify the linearizer; usually will be the same as the PTC selection.
- `export SELECTION_PTC_BFK_V=$SELECTION_PTC_V`: The selection of raws to verify the brighter-fatter kernel; usually will be the same as the PTC selection.
- `export SELECTION_FLAT_g_V=ïnstrument='LSSTComCam and selection_stringë`: The selection of raws to verify the g-band flat. There will need to be one selection for each ugrizy.

# 7 Full Calibration Example

This section contains the commands used to generate, certify, chain, and verify calibrations for LSSTComCam DP1 processing. This is adapted from the script attached to DM-48520.

Note that this is not a complete construction of all calibrations; for this we used pre-existing linearity, ptc, and bfk.

In addition to the calibration building, this includes some checkpoints to download mosaic images for visual spot-checking to ensure that nothing has gone terribly wrong before continuing on all the way to verification reports.

```
export USER_CALIB_PREFIX=""
export INSTRUMENT=LSSTComCam
```

```
export TICKET=DM-48520
export REPO=/repo/main
export RAW_COLLECTION=LSSTComCam/raw/all
export CALIB_COLLECTIONS="LSSTComCam/calib/DM-48650,\
    LSSTComCam/calib/DM-47447/gainFixup/ptc.20241107a,\
    LSSTComCam/calib/DM-46360/isrTaskLSST/linearizer.20240926a,\
    LSSTComCam/calib/DM-46360/isrTaskLSST/bfk.20240926a,\
    LSSTComCam/calib/DM-46360/isrTaskLSST/ptc.20240926a"
export TAG=DP1
export RERUN=20250207a
export BOOTSTRAP_RUN_NUMBER=1

# These are certification environment variables.
export START_DATE=1970-01-01T00:00:00
export CERT_RERUN=20250207a

export SELECTION_BIAS="instrument='$INSTRUMENT' and \
    ((exposure.day_obs=20241208 and exposure.seq_num >= 696) or \
    (exposure.day_obs=20241206 and exposure.seq_num >=427)) and \
    exposure.observation_type='bias'"
export SELECTION_DARK="instrument='$INSTRUMENT' and \
    ((exposure.day_obs=20241208 and exposure.seq_num >= 696) or \
    (exposure.day_obs=20241206 and exposure.seq_num >=427)) and \
    exposure.observation_type='dark'"
export SELECTION_FLAT_BOOTSTRAP="instrument='$INSTRUMENT' and \
    exposure.day_obs=20240806 and exposure.science_program='BLOCK-T68' and \
    exposure.observation_reason='flat' and physical_filter='r_03'"

export SELECTION_PTC_CTI="instrument='$INSTRUMENT' and \
    exposure.day_obs=20240806 and \
    and exposure.science_program='BLOCK-T68' and \
    exposure.observation_reason='comcam_ptc' and \
    physical_filter='r_03'"

export SELECTION_FLAT_u="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241110 and exposure.seq_num in (44..63)"
export SELECTION_FLAT_g="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241126 and exposure.seq_num in (21..40)"
export SELECTION_FLAT_r="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241111 and exposure.seq_num in (17..36)"
export SELECTION_FLAT_i="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241120 and exposure.seq_num in (41..64) and exposure.seq_num != 58"
export SELECTION_FLAT_z="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241112 and exposure.seq_num in (28..48)"
```

```
export SELECTION_FLAT_y="instrument='$INSTRUMENT' and \
    exposure.day_obs=20241119 and exposure.seq_num in (21..36)"

cd /sdf/data/rubin/user/erykoff/calibrations/LSSTComCam/DM-48520
mkdir -p images

# This is a quick check that the filled-in template looks
# correct (watch for missing env vars!)
cat $CP_PIPE_DIR/bps/templates/bps_biasBootstrap.yaml | envsubst

bps submit $CP_PIPE_DIR/bps/templates/bps_biasBootstrap.yaml

# Download a couple of images to check them.
butler retrieve-artifacts $REPO images \
    -d biasBootstrap \
   --collections ${USER_CALIB_PREFIX}$INSTRUMENT/calib/$TICKET/$TAG/biasBootstrapGen.$RERUN/run${BOOTSTRAP_RUN_NUMBER} \
    --where "instrument='${INSTRUMENT}' and detector in (5, 8)" --no-preserve-path

bps submit $CP_PIPE_DIR/bps/templates/bps_darkBootstrap.yaml

butler retrieve-artifacts $REPO images \
    -d darkBootstrap \
   --collections ${USER_CALIB_PREFIX}$INSTRUMENT/calib/$TICKET/$TAG/darkBootstrapGen.$RERUN/run${BOOTSTRAP_RUN_NUMBER} \
    --where "instrument='${INSTRUMENT}' and detector in (5, 8)" --no-preserve-path

bps submit $CP_PIPE_DIR/bps/templates/bps_flatBootstrap.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_defects.yaml

# We are certifying (applying a date range) and chaining the calibrations
# as we go, which makes downstream processing easier without having to
# track many collections.

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/defectGen.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/defects.${CERT_RERUN} \
    defects

butler collection-chain $REPO \
    --mode redefine \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/defects.${CERT_RERUN}
```

```
# The gainFixup PTC which is appropriate for the on-sky data after the
# ComCam warmup (when it was installed) is certified *after* the lab data.
# However, the lab PTC in LSSTComCam/calib/DM-46360/isrTaskLSST/ptc.20240926a
# is possibly more appropriate for the CTI calibration from the same
# dataset before the gains changed at the 0.3-0.5% level.

cat $CP_PIPE_DIR/bps/templates/bps_cti.yaml | envsubst

bps submit $CP_PIPE_DIR/bps/templates/bps_cti.yaml

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/ctiGen.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/cti.${CERT_RERUN} \
    cti

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/cti.${CERT_RERUN}

cat $CP_PIPE_DIR/bps/templates/bps_bias.yaml | envsubst

bps submit $CP_PIPE_DIR/bps/templates/bps_bias.yaml

butler retrieve-artifacts $REPO images \
    -d biasMosaic8 \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/biasGen.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/biasGen.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/bias.${CERT_RERUN} \
    bias

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/bias.${CERT_RERUN}

cat $CP_PIPE_DIR/bps/templates/bps_dark.yaml | envsubst

bps submit $CP_PIPE_DIR/bps/templates/bps_dark.yaml
```

```
butler retrieve-artifacts $REPO images \
    -d darkMosaic8 \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/darkGen.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/darkGen.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/dark.${CERT_RERUN} \
    dark

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/dark.${CERT_RERUN}

cat $CP_PIPE_DIR/bps/templates/bps_flat_u.yaml | envsubst

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_u.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_g.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_r.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_i.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_z.yaml

bps submit $CP_PIPE_DIR/bps/templates/bps_flat_y.yaml

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-u.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-g.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-r.${RERUN} \
```

```
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-i.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-z.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler retrieve-artifacts $REPO images \
    -d flat \
    --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-y.${RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-u.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-u.${CERT_RERUN} \
    flat

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-u.${CERT_RERUN}

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-g.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-g.${CERT_RERUN} \
    flat

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-g.${CERT_RERUN}

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-r.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-r.${CERT_RERUN} \
    flat
```

```
butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-r.${CERT_RERUN}

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-i.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-i.${CERT_RERUN} \
    flat

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-i.${CERT_RERUN}

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-z.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-z.${CERT_RERUN} \
    flat

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-z.${CERT_RERUN}

butler certify-calibrations $REPO \
    --begin-date $START_DATE \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-y.${RERUN} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-y.${CERT_RERUN} \
    flat

butler collection-chain $REPO \
    --mode prepend \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET} \
    ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flat-y.${CERT_RERUN}

############################################################
## Verification section.
############################################################

export VERIFY_RERUN=20240210a
```

```
export SELECTION_SCIENCE_V="instrument='$INSTRUMENT' and \
    exposure in (2024120100160,2024120700307,2024120700297,2024120300156,\
    2024120700339,2024112000223)"
# Other exports
export SELECTION_BIAS_V="instrument='$INSTRUMENT' and \
    exposure in (2024102100509,2024102600031,2024102900175,2024110100164,\
    2024110800008,2024111200387,2024111600316,2024112200390,2024112300315,\
    2024112400006,2024112700443,2024120500004,2024120500266,2024120500272,\
    2024120500291,2024120700571,2024120700576,2024120700592,2024120700598,\
    2024120800704,2024120800724,2024120900006,2024120900284,2024121000596,\
    2024121100007)"
export SELECTION_DARK_V="instrument='$INSTRUMENT' and \
    exposure in (2024102300019,2024102300035,2024102500019,2024102500044,\
    2024103000012,2024103000037,2024110100149,2024110100175,2024110500279,\
    2024110500301,2024111100307,2024111100334,2024111600322,2024111600350,\
    2024112000330,2024112000355,2024112700437,2024112700455,2024120100481,\
    2024120100507,2024120700582,2024120700588,2024120700604,2024120700610,\
    2024121000007,2024121000590,2024121000612)"
export SELECTION_FLAT_u_V=$SELECTION_FLAT_u
export SELECTION_FLAT_g_V=$SELECTION_FLAT_g
export SELECTION_FLAT_r_V=$SELECTION_FLAT_r
export SELECTION_FLAT_i_V=$SELECTION_FLAT_i
export SELECTION_FLAT_z_V=$SELECTION_FLAT_z
export SELECTION_FLAT_y_V=$SELECTION_FLAT_y

# Run science frames.

cat $CP_VERIFY_DIR/bps/templates/bps_verify_science.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_science.yaml

butler retrieve-artifacts $REPO images \
    -d verifyScience8 \
   --collections ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyScience.${VERIFY_RERUN} \
    --where "instrument='${INSTRUMENT}'" --no-preserve-path --find-first

# Other verifications.

cat $CP_VERIFY_DIR/bps/templates/bps_verify_bias.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_bias.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_dark.yaml | envsubst
```

```
bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_dark.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_u.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_u.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_g.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_g.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_r.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_r.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_i.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_i.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_z.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_z.yaml

cat $CP_VERIFY_DIR/bps/templates/bps_verify_flat_y.yaml | envsubst

bps submit $CP_VERIFY_DIR/bps/templates/bps_verify_flat_y.yaml

# Make TAXICAB reports.
# Note that the order matters, and should be verify runs first,
# followed by gen runs.

$CP_VERIFY_DIR/bin/cpv_report.py -r $REPO \
    -O ~/public_html/cpv_reports/TAXICAB-23 \
    --do_overwrite \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyBias.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyDark.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-u.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-g.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-r.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-i.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-z.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/verifyFlat-y.${VERIFY_RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/biasGen.${RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/darkGen.${RERUN} \
    -c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-u.${RERUN} \
```

```
-c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-g.${RERUN} \
-c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-r.${RERUN} \
-c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-i.${RERUN} \
-c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-z.${RERUN} \
-c ${USER_CALIB_PREFIX}${INSTRUMENT}/calib/${TICKET}/${TAG}/flatGen-y.${RERUN}
```

# 8 Conclusions

# A References

# B Acronyms

| Acronym | Description |
|---|---|
| BPS | Batch Production Service |
| BTS | Base (La Serena) Test Stand |
| CP | Contingency Planning |
| CTI | Charge Transfer Inefficiency |
| DM | Data Management |
| DMTN | DM Technical Note |
| DP1 | Data Preview 1 |
| FGCM | Forward Global Calibration Model |
| HTML | HyperText Markup Language |
| ISO | Information Security Officer |
| LATISS | LSST Atmospheric Transmission Imager and Slitless Spectrograph |
| LOVE | LSST Operators Visualization Environment |
| PTC | Photon Transfer Curve |
| S3 | (Amazon) Simple Storage Service |
| SLAC | SLAC National Accelerator Laboratory |
| TAI | International Atomic Time |
| TAXICAB | Telescope and Auxiliary Instrumentation Calibration Acceptance Board |
| TBD | To Be Defined (Determined) |

| | |
|---|---|
| TTS | Tucson Test Stand |
| US | United States |
| USDF | United States Data Facility |
| bps | bit(s) per second |