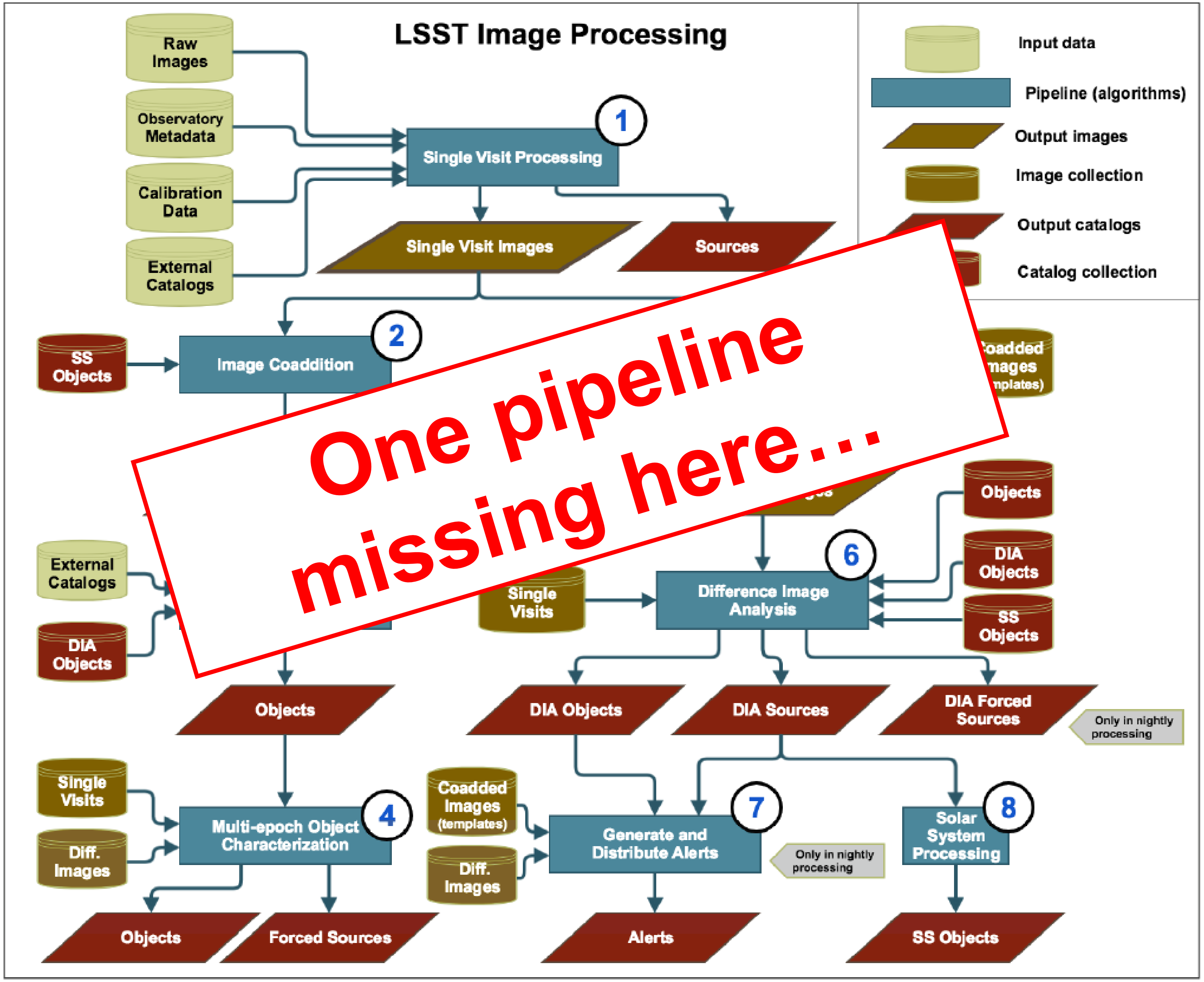
Calibrations: production, validation and use

Thibault Guillemin

Focal plane commissioning workshop

March 27-29, 2029

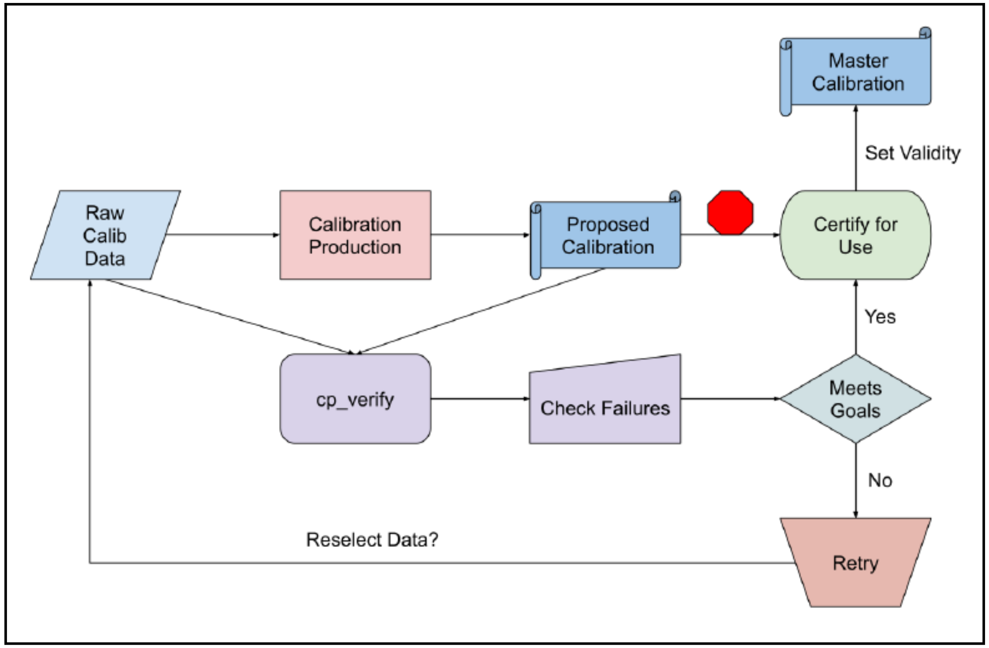
Overall view

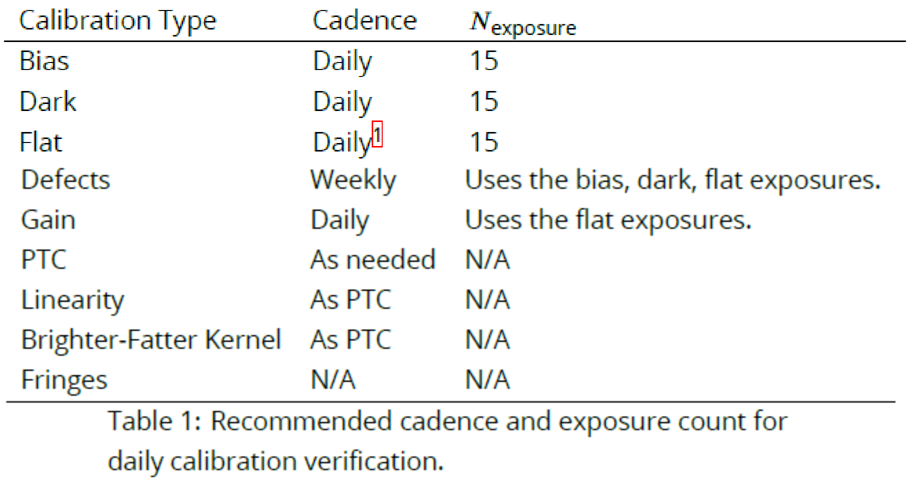


One pipeline missing here…

Calibration pipeline

Document: https://dmtn-222.lsst.io/v/u-czw-20220321/index.html





Calibration Type	Cadence	N_{exposure}
Bias	Daily	15
Dark	Daily	15
Flat	Daily[1]	15
Defects	Weekly	Uses the bias, dark, flat exposures.
Gain	Daily	Uses the flat exposures.
PTC	As needed	N/A
Linearity	As PTC	N/A
Brighter-Fatter Kernel	As PTC	N/A
Fringes	N/A	N/A

Table 1: Recommended cadence and exposure count for daily calibration verification.

Calibration exposures and calibration products

- Calibration exposures:

1) **Bias**: shutter off, no exposure time
➔ Electronic offset for an unexposed pixel, readout noise

2) **Dark**: shutter off, exposure time of various duration
➔ Readout noise, dark current, subtle readout instabilities

3) **Flat**: uniform exposure of known flux (usually taken in pairs)
➔ Gain, full well, linearity, Brighter-Fatter
- PTC (Photon Transfer Curve): variance of uniform images as a function of their average (increasing flux)
- Flats versus λ: to measure the quantum efficiency versus lambda

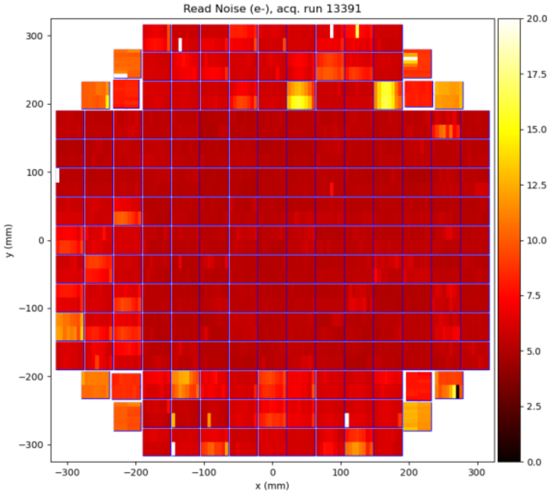
- Calibration products

- Combined bias: average bias image build from n bias input exposures
- Defects: cold and hot pixels from flats and darks

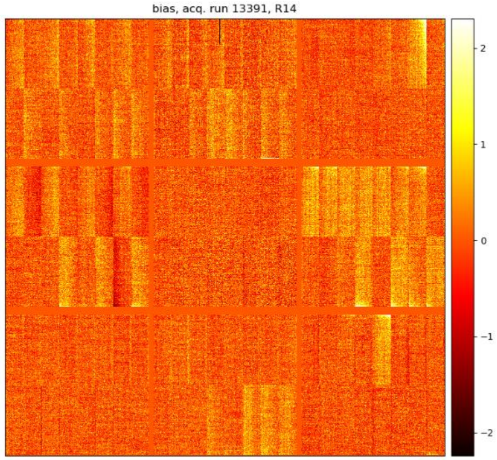
eo_pipe: camera tool

https://s3df.slac.stanford.edu/data/rubin/lsstcam/13391/

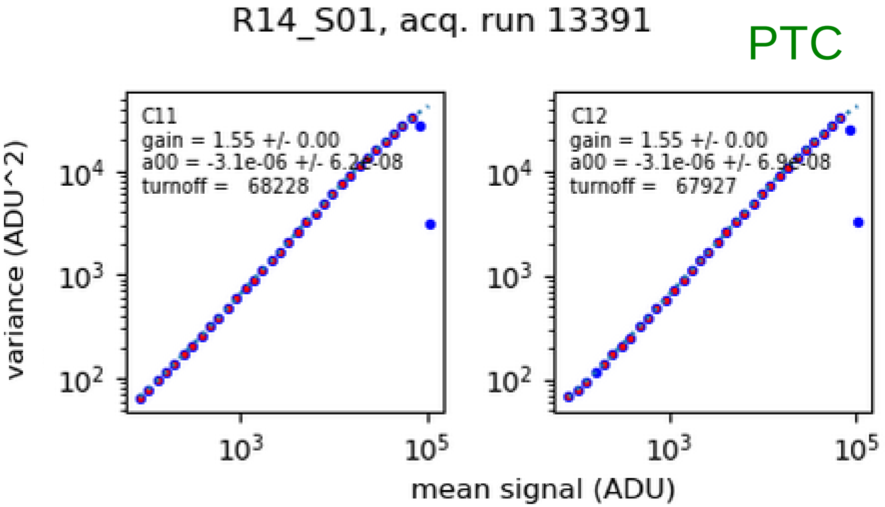
Read noise



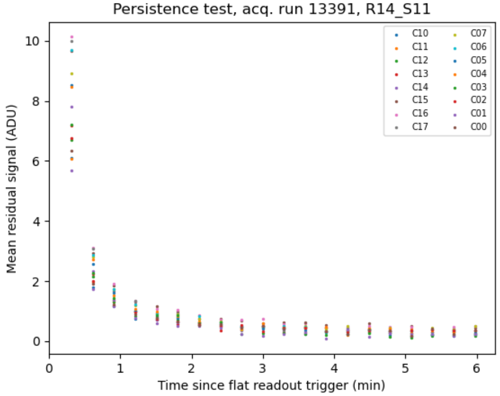
Combined bias



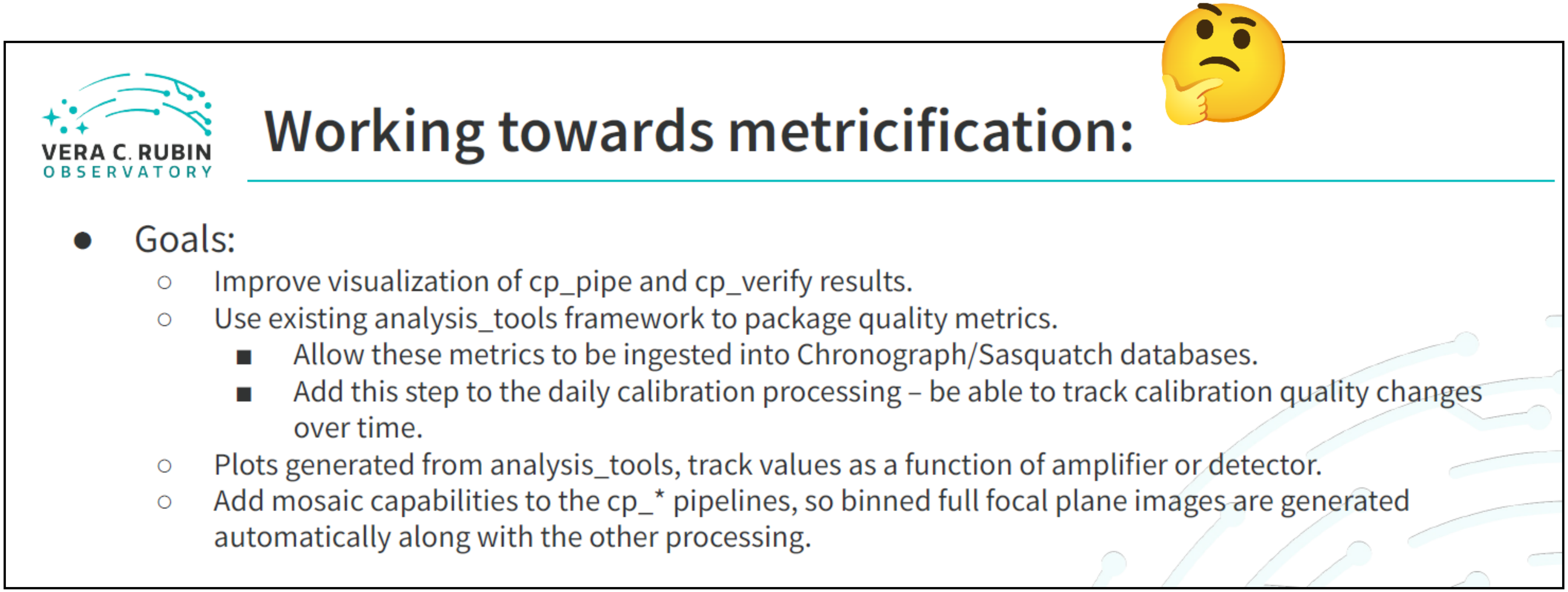
PTC



Persistence



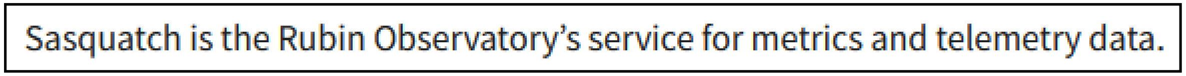
Calibration monitoring/validation: DM efforts (1/2)



Working towards metricification:

- Goals:
  - Improve visualization of cp_pipe and cp_verify results.
  - Use existing analysis_tools framework to package quality metrics.
    - Allow these metrics to be ingested into Chronograph/Sasquatch databases.
    - Add this step to the daily calibration processing – be able to track calibration quality changes over time.
  - Plots generated from analysis_tools, track values as a function of amplifier or detector.
  - Add mosaic capabilities to the cp_* pipelines, so binned full focal plane images are generated automatically along with the other processing.

From C. Waters



Sasquatch is the Rubin Observatory's service for metrics and telemetry data.

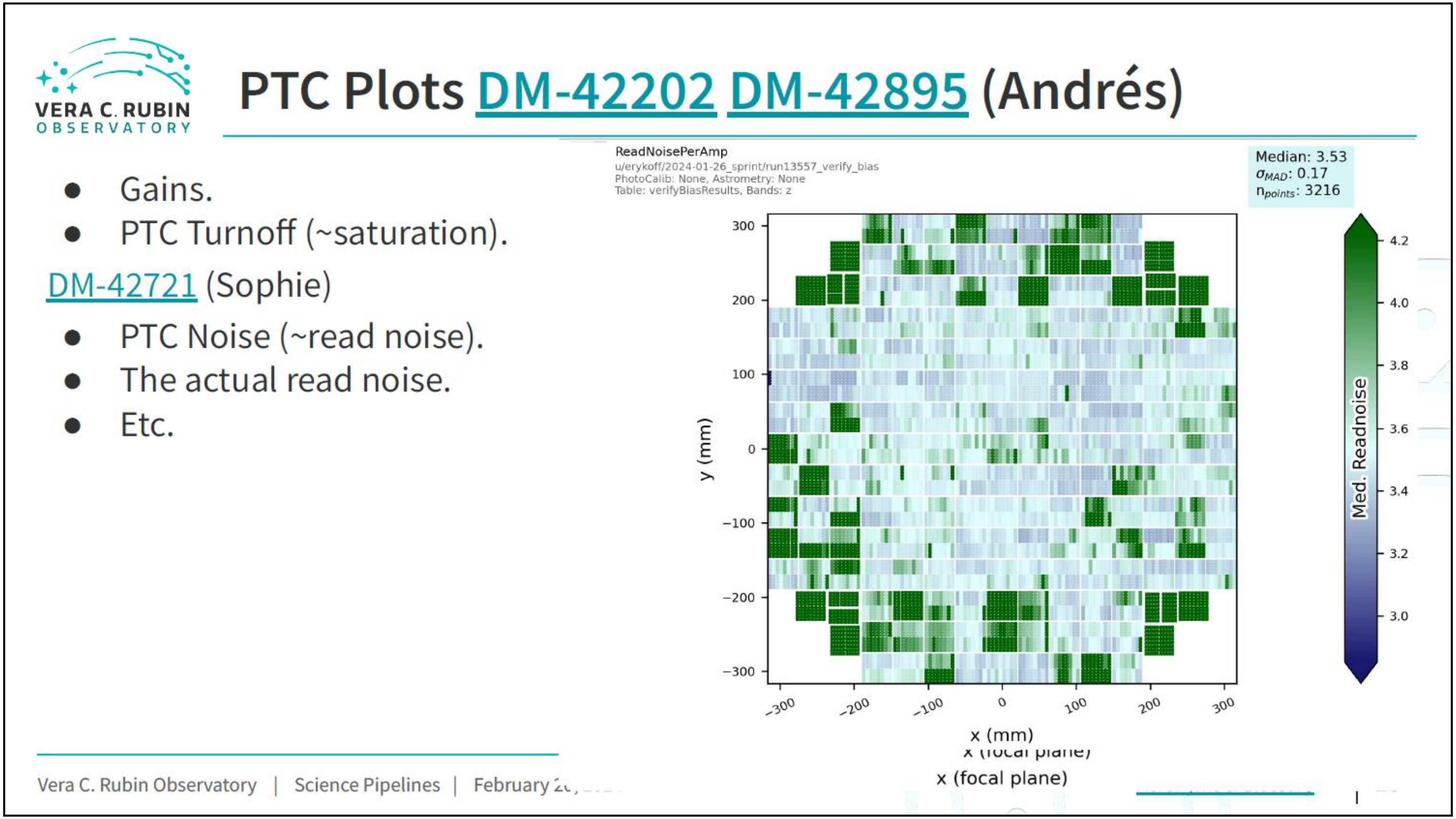
Framework based on:
- analysis_tools
- plot_navigator

In development…A lot of discussions ongoing…

TAXICAB: Telescope and AuXiliary Instrumentation Calibration Acceptance Board

Calibration monitoring/validation: DM efforts (2/2)

From C. Waters



PTC Plots DM-42202 DM-42895 (Andrés)

- Gains.
- PTC Turnoff (~saturation).

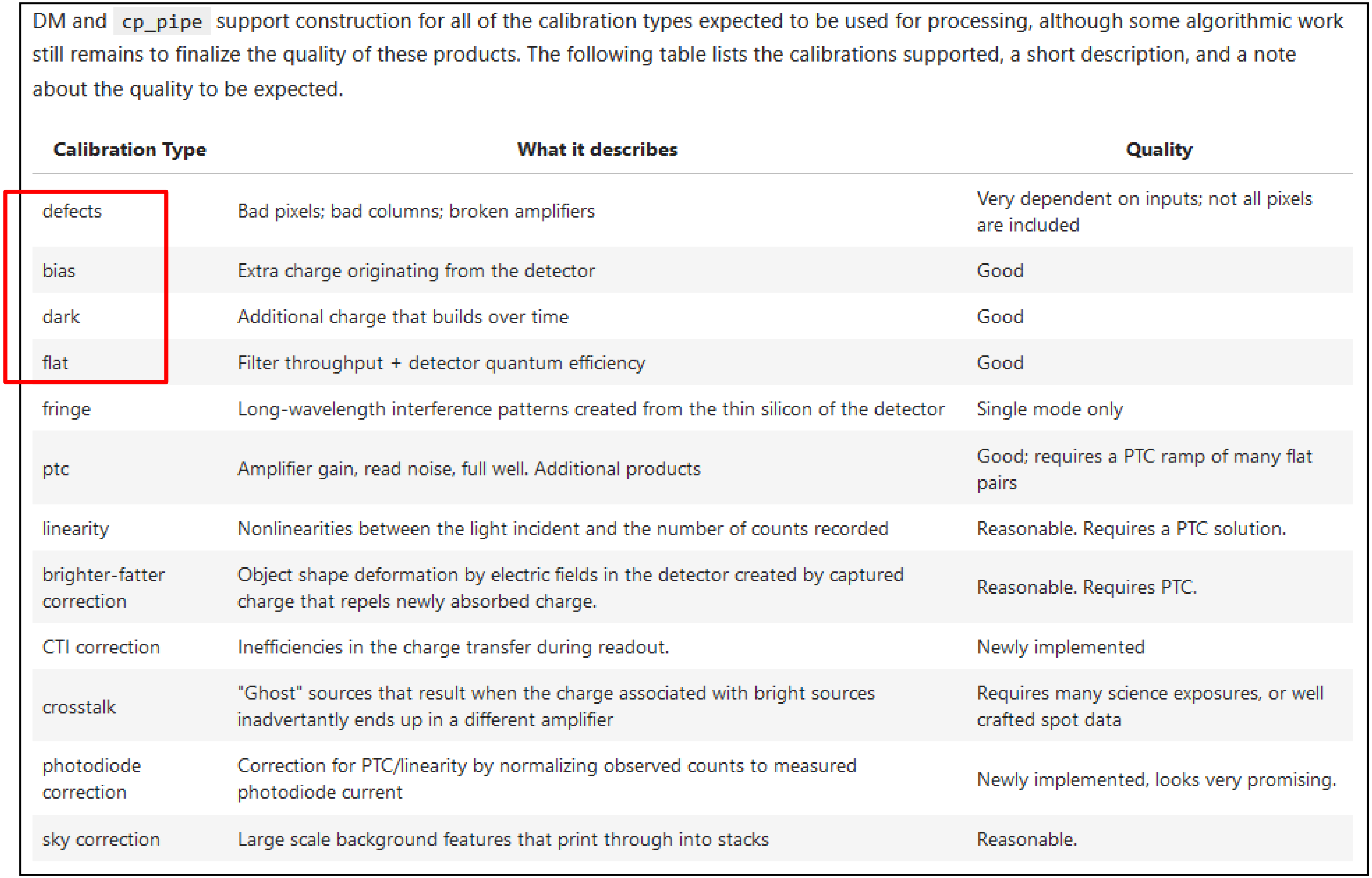
DM-42721 (Sophie)

- PTC Noise (~read noise).
- The actual read noise.
- Etc.

Vera C. Rubin Observatory | Science Pipelines | February 2…

https://usdf-rsp.slac.stanford.edu/plot-navigator/dashboard_gen3

Calibration products

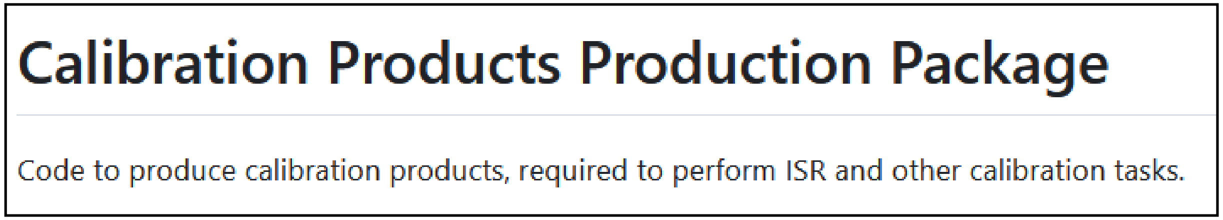


DM and `cp_pipe` support construction for all of the calibration types expected to be used for processing, although some algorithmic work still remains to finalize the quality of these products. The following table lists the calibrations supported, a short description, and a note about the quality to be expected.

Calibration Type	What it describes	Quality
defects	Bad pixels; bad columns; broken amplifiers	Very dependent on inputs; not all pixels are included
bias	Extra charge originating from the detector	Good
dark	Additional charge that builds over time	Good
flat	Filter throughput + detector quantum efficiency	Good
fringe	Long-wavelength interference patterns created from the thin silicon of the detector	Single mode only
ptc	Amplifier gain, read noise, full well. Additional products	Good; requires a PTC ramp of many flat pairs
linearity	Nonlinearities between the light incident and the number of counts recorded	Reasonable. Requires a PTC solution.
brighter-fatter correction	Object shape deformation by electric fields in the detector created by captured charge that repels newly absorbed charge.	Reasonable. Requires PTC.
CTI correction	Inefficiencies in the charge transfer during readout.	Newly implemented
crosstalk	"Ghost" sources that result when the charge associated with bright sources inadvertantly ends up in a different amplifier	Requires many science exposures, or well crafted spot data
photodiode correction	Correction for PTC/linearity by normalizing observed counts to measured photodiode current	Newly implemented, looks very promising.
sky correction	Large scale background features that print through into stacks	Reasonable.

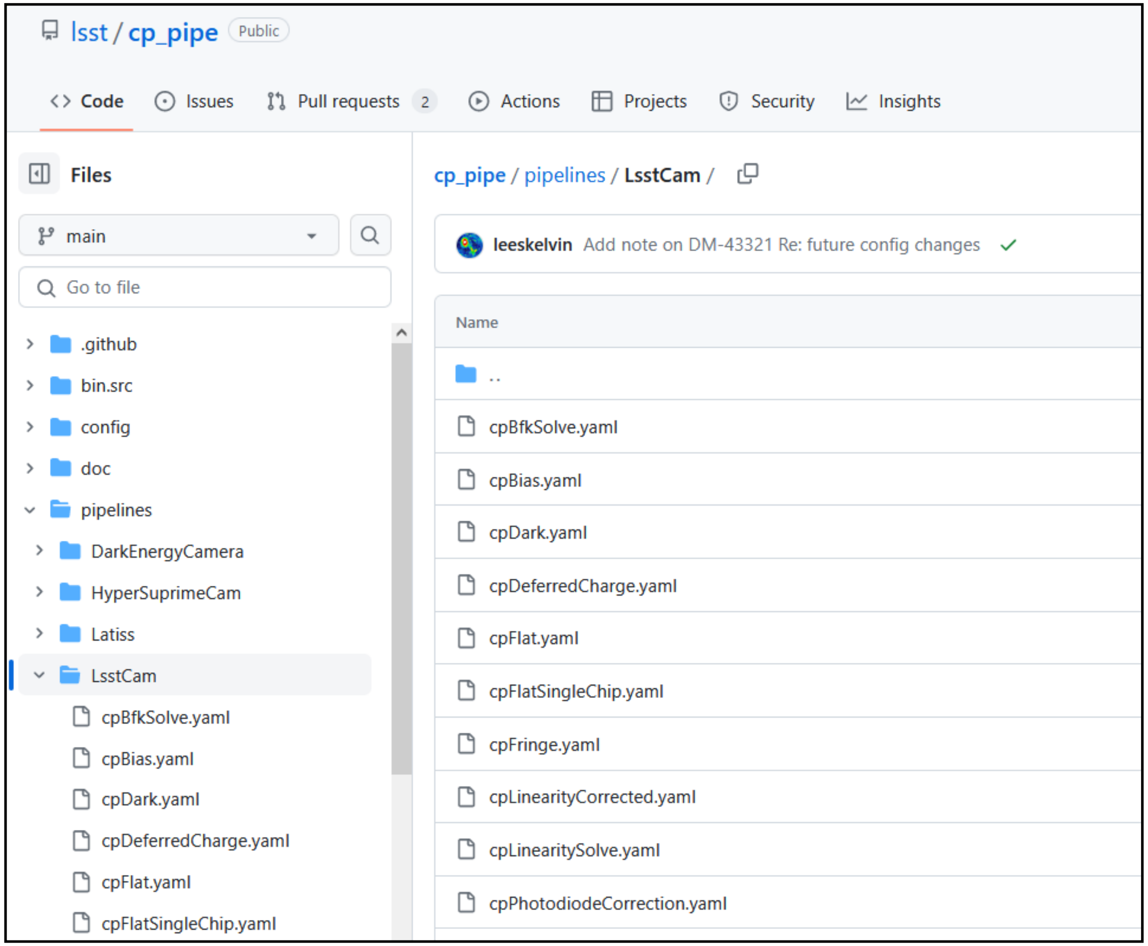
From C. Waters

cp_pipe



Calibration Products Production Package

Code to produce calibration products, required to perform ISR and other calibration tasks.



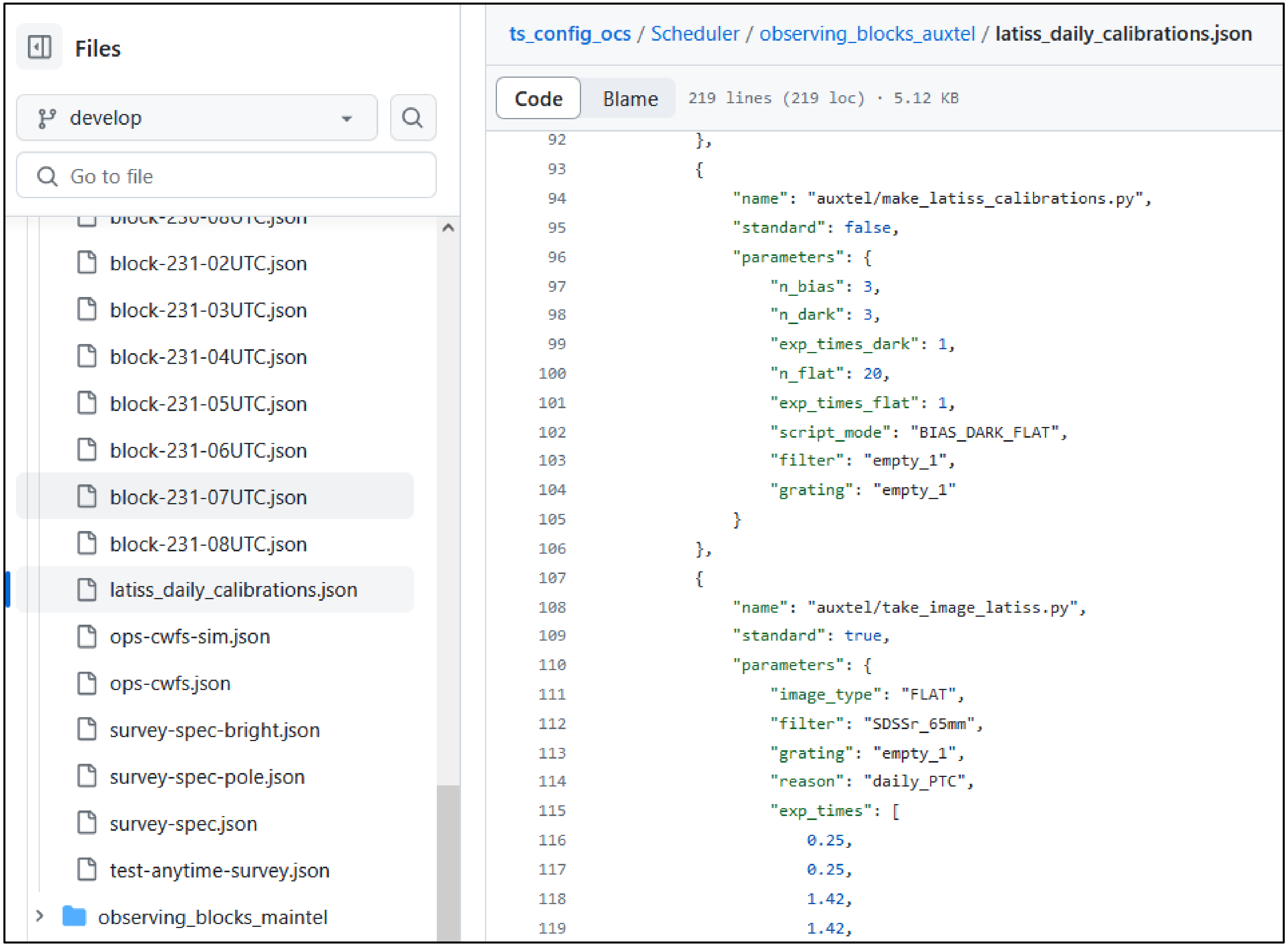
cp_pipe / pipelines / LsstCam /

leeskelvin Add note on DM-43321 Re: future config changes ✓

- .github
- bin.src
- config
- doc
- pipelines
  - DarkEnergyCamera
  - HyperSuprimeCam
  - Latiss
  - LsstCam
    - cpBfkSolve.yaml
    - cpBias.yaml
    - cpDark.yaml
    - cpDeferredCharge.yaml
    - cpFlat.yaml
    - cpFlatSingleChip.yaml

Name
..
cpBfkSolve.yaml
cpBias.yaml
cpDark.yaml
cpDeferredCharge.yaml
cpFlat.yaml
cpFlatSingleChip.yaml
cpFringe.yaml
cpLinearityCorrected.yaml
cpLinearitySolve.yaml
cpPhotodiodeCorrection.yaml

AuxTel daily calibrations



- block-230-08UTC.json
- block-231-02UTC.json
- block-231-03UTC.json
- block-231-04UTC.json
- block-231-05UTC.json
- block-231-06UTC.json
- block-231-07UTC.json
- block-231-08UTC.json
- latiss_daily_calibrations.json
- ops-cwfs-sim.json
- ops-cwfs.json
- survey-spec-bright.json
- survey-spec-pole.json
- survey-spec.json
- test-anytime-survey.json
- observing_blocks_maintel

ts_config_ocs / Scheduler / observing_blocks_auxtel / latiss_daily_calibrations.json

219 lines (219 loc) · 5.12 KB

```
            },
            {
                "name": "auxtel/make_latiss_calibrations.py",
                "standard": false,
                "parameters": {
                    "n_bias": 3,
                    "n_dark": 3,
                    "exp_times_dark": 1,
                    "n_flat": 20,
                    "exp_times_flat": 1,
                    "script_mode": "BIAS_DARK_FLAT",
                    "filter": "empty_1",
                    "grating": "empty_1"
                }
            },
            {
                "name": "auxtel/take_image_latiss.py",
                "standard": true,
                "parameters": {
                    "image_type": "FLAT",
                    "filter": "SDSSr_65mm",
                    "grating": "empty_1",
                    "reason": "daily_PTC",
                    "exp_times": [
                        0.25,
                        0.25,
                        1.42,
                        1.42,
```

Questions

- How to list from the butler?
- How to access (plots, metrics)?
- How to know which calibrations were or are used?
- How to apply specific collections in pipetask?
- How to create calibrations?
- How to validate calibrations?
- How to certify in the butler?
- How to export and import in the butler?

Questions

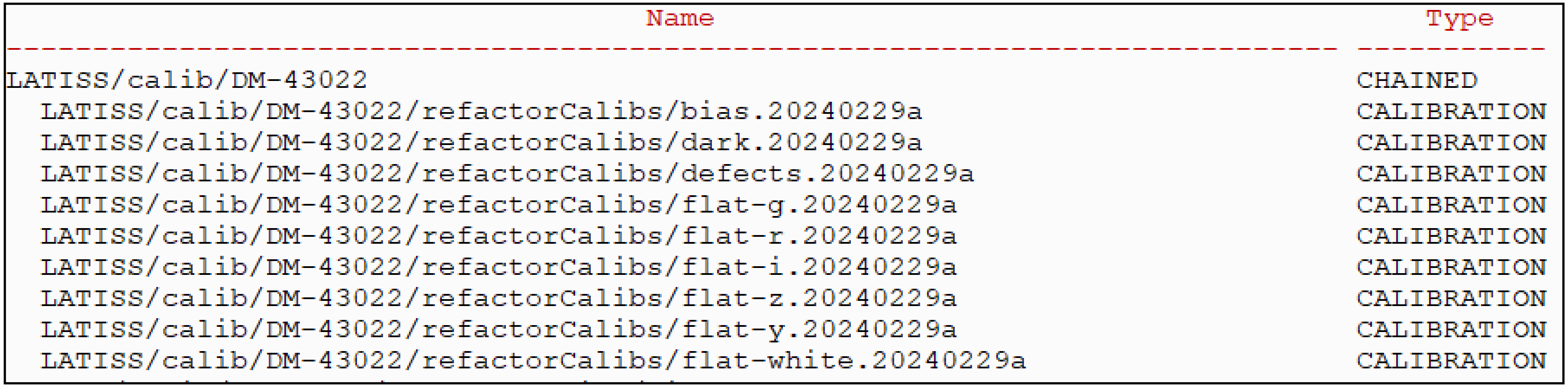
- How to list from the butler?
- How to access (plots, metrics)?
- How to know which calibrations were or are used?
- How to apply specific collections in pipetask?
- How to create calibrations?
- How to validate calibrations?
- How to certify in the butler?
- How to export and import in the butler?

Calibration collections

butler: /sps/lsst/groups/FocalPlane/SLAC/run6/butler/test_auxtel/main_240311

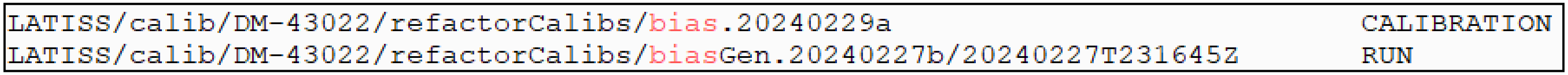
butler query-collections $REPO

CHAINED



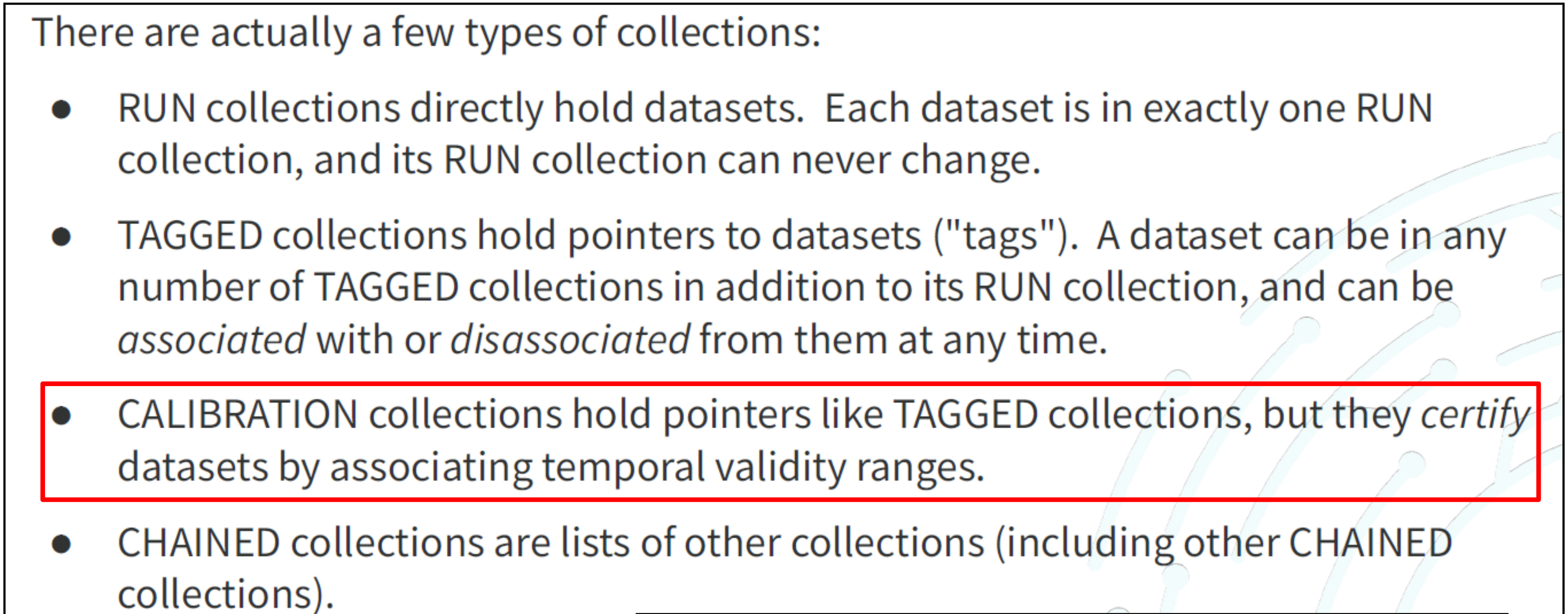
```
                                     Name                                              Type
---------------------------------------------------------------------------------  -----------
LATISS/calib/DM-43022                                                              CHAINED
  LATISS/calib/DM-43022/refactorCalibs/bias.20240229a                              CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/dark.20240229a                              CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/defects.20240229a                           CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-g.20240229a                            CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-r.20240229a                            CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-i.20240229a                            CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-z.20240229a                            CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-y.20240229a                            CALIBRATION
  LATISS/calib/DM-43022/refactorCalibs/flat-white.20240229a                        CALIBRATION
```

bias case



```
LATISS/calib/DM-43022/refactorCalibs/bias.20240229a                                CALIBRATION
LATISS/calib/DM-43022/refactorCalibs/biasGen.20240227b/20240227T231645Z            RUN
```

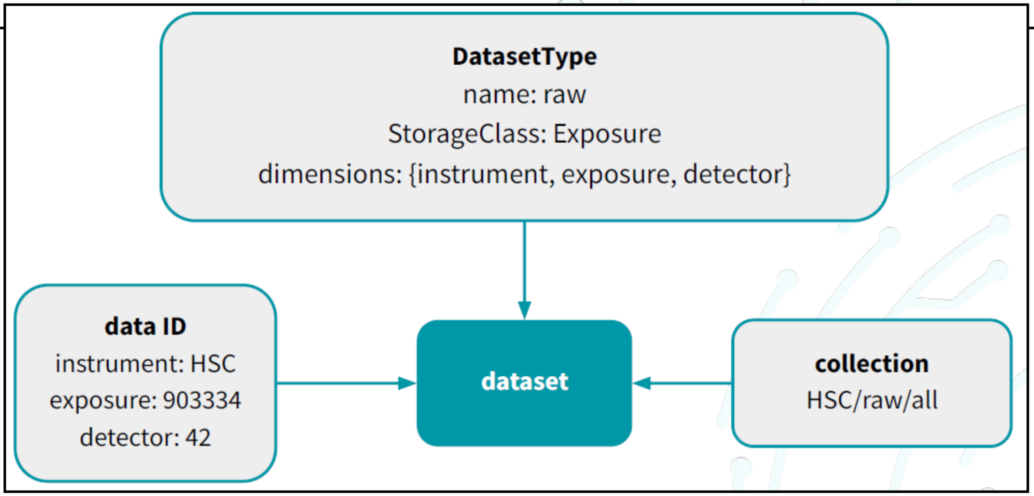
Types of collections



There are actually a few types of collections:

- RUN collections directly hold datasets. Each dataset is in exactly one RUN collection, and its RUN collection can never change.
- TAGGED collections hold pointers to datasets ("tags"). A dataset can be in any number of TAGGED collections in addition to its RUN collection, and can be *associated* with or *disassociated* from them at any time.
- CALIBRATION collections hold pointers like TAGGED collections, but they *certify* datasets by associating temporal validity ranges.
- CHAINED collections are lists of other collections (including other CHAINED collections).

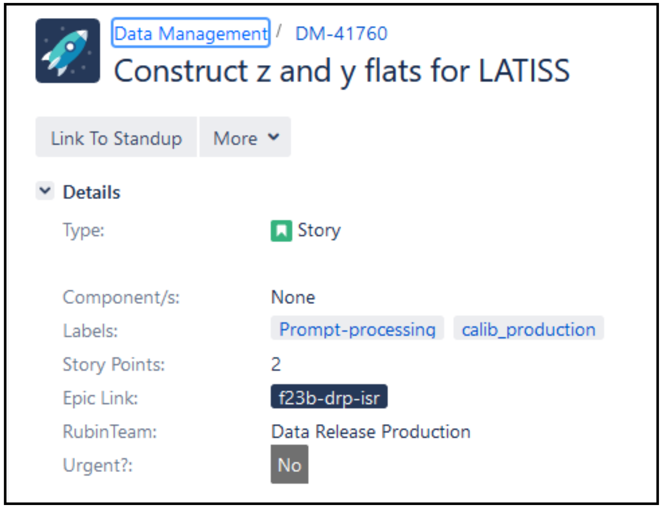
From J. Bosh



Chaining calibration collections

USDF
Incomplete CHAINED collection

```
LATISS/calib/DM-41760                                              CHAINED
  LATISS/calib/DM-41760/flat-z                                     CALIBRATION
  LATISS/calib/DM-41760/flat-y                                     CALIBRATION
```



Data Management / DM-41760

Construct z and y flats for LATISS

Details

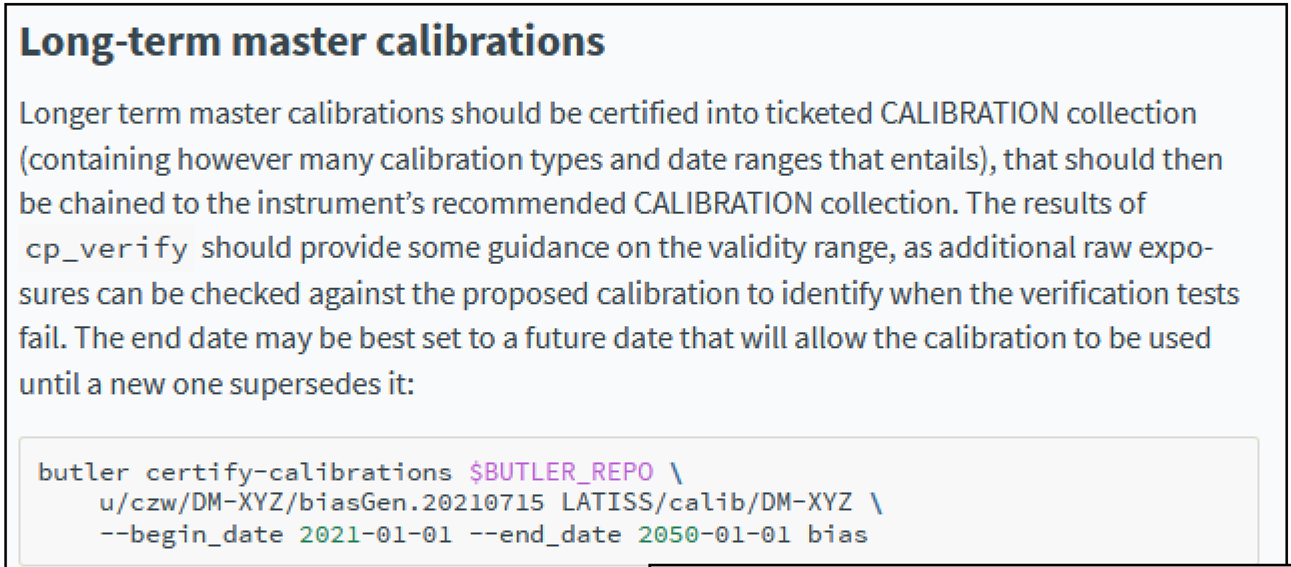
Type:	Story
Component/s:	None
Labels:	Prompt-processing calib_production
Story Points:	2
Epic Link:	f23b-drp-isr
RubinTeam:	Data Release Production
Urgent?:	No

Useful to have a look to the associated JIRA ticket

```
# Chain the ticket collections together into a bundle.
butler collection-chain $REPO LATISS/calib/$TICKET \
       LATISS/calib/$TICKET/flat-z LATISS/calib/$TICKET/flat-y

# Chain the bundle to the top level calib collection.
butler collection-chain --mode prepend $REPO LATISS/calib \
       LATISS/calib/$TICKET  ...
```

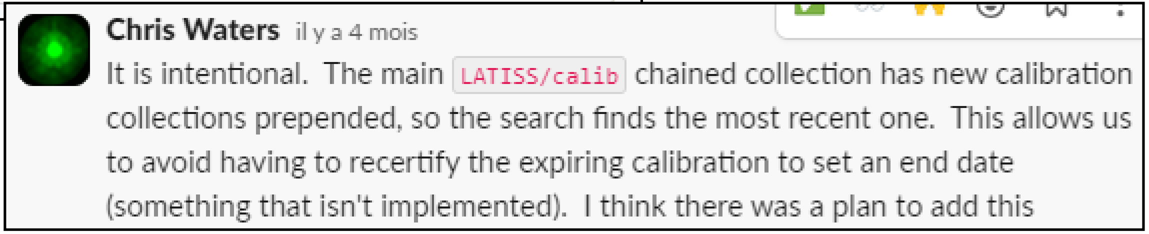
Certification and validity ranges



Long-term master calibrations

Longer term master calibrations should be certified into ticketed CALIBRATION collection (containing however many calibration types and date ranges that entails), that should then be chained to the instrument's recommended CALIBRATION collection. The results of `cp_verify` should provide some guidance on the validity range, as additional raw exposures can be checked against the proposed calibration to identify when the verification tests fail. The end date may be best set to a future date that will allow the calibration to be used until a new one supersedes it:

```
butler certify-calibrations $BUTLER_REPO \
    u/czw/DM-XYZ/biasGen.20210715 LATISS/calib/DM-XYZ \
    --begin_date 2021-01-01 --end_date 2050-01-01 bias
```



Chris Waters il y a 4 mois

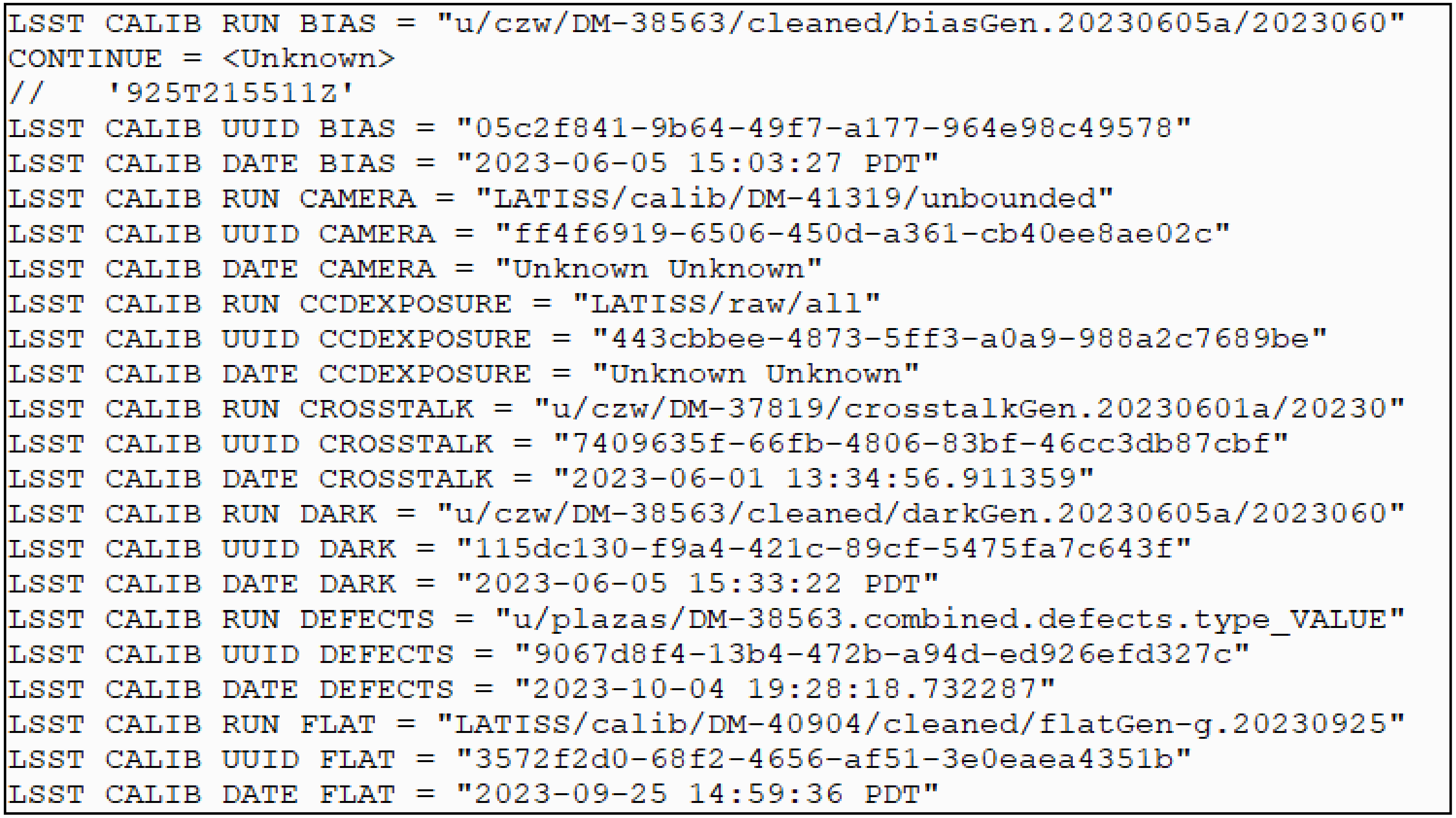
It is intentional. The main `LATISS/calib` chained collection has new calibration collections prepended, so the search finds the most recent one. This allows us to avoid having to recertify the expiring calibration to set an end date (something that isn't implemented). I think there was a plan to add this

Getting the timespan from:
butler.registry.queryDatasetAssociations ('bias','LATISS/calib')

```
[2020-01-01T00:00:00, 2050-01-01T00:00:00) u/czw/DM-28920/biasGen.20210702a/20210702T215049Z
[2022-10-05T00:00:00, ∞) LATISS/calib/DM-36484/biasGen.20221005a/20221006T000101Z
[2022-11-03T00:00:00, ∞) LATISS/calib/DM-36719/biasGen.20221107b/20221107T213306Z
[2022-11-03T00:00:00, ∞) u/czw/DM-37811/parOStest.20230202a/biasGen.20230202a/20230202T235503Z
[2023-04-25T00:00:00, ∞) LATISS/calib/DM-38946/noRGseq/biasGen.20230428a/20230428T210637Z
[2023-04-25T00:00:00, ∞) u/czw/DM-38563/cleaned/biasGen.20230605a/20230605T215546Z
[2023-11-01T00:00:00, ∞) LATISS/calib/DM-43022/refactorCalibs/biasGen.20240227b/20240227T231645Z
```

Reading calibrations from exposure metadata

```
expMetadata = butler.get('postISRCCD.metadata',instrument='LATISS', detector=0,
exposure=2023082900412, collections=['LATISS/runs/
AUXTEL_DRP_IMAGING_20230509_20231207/w_2023_49/PREOPS-4648'])
```



```
LSST CALIB RUN BIAS = "u/czw/DM-38563/cleaned/biasGen.20230605a/2023060"
CONTINUE = <Unknown>
//    '925T215511Z'
LSST CALIB UUID BIAS = "05c2f841-9b64-49f7-a177-964e98c49578"
LSST CALIB DATE BIAS = "2023-06-05 15:03:27 PDT"
LSST CALIB RUN CAMERA = "LATISS/calib/DM-41319/unbounded"
LSST CALIB UUID CAMERA = "ff4f6919-6506-450d-a361-cb40ee8ae02c"
LSST CALIB DATE CAMERA = "Unknown Unknown"
LSST CALIB RUN CCDEXPOSURE = "LATISS/raw/all"
LSST CALIB UUID CCDEXPOSURE = "443cbbee-4873-5ff3-a0a9-988a2c7689be"
LSST CALIB DATE CCDEXPOSURE = "Unknown Unknown"
LSST CALIB RUN CROSSTALK = "u/czw/DM-37819/crosstalkGen.20230601a/20230"
LSST CALIB UUID CROSSTALK = "7409635f-66fb-4806-83bf-46cc3db87cbf"
LSST CALIB DATE CROSSTALK = "2023-06-01 13:34:56.911359"
LSST CALIB RUN DARK = "u/czw/DM-38563/cleaned/darkGen.20230605a/2023060"
LSST CALIB UUID DARK = "115dc130-f9a4-421c-89cf-5475fa7c643f"
LSST CALIB DATE DARK = "2023-06-05 15:33:22 PDT"
LSST CALIB RUN DEFECTS = "u/plazas/DM-38563.combined.defects.type_VALUE"
LSST CALIB UUID DEFECTS = "9067d8f4-13b4-472b-a94d-ed926efd327c"
LSST CALIB DATE DEFECTS = "2023-10-04 19:28:18.732287"
LSST CALIB RUN FLAT = "LATISS/calib/DM-40904/cleaned/flatGen-g.20230925"
LSST CALIB UUID FLAT = "3572f2d0-68f2-4656-af51-3e0eaea4351b"
LSST CALIB DATE FLAT = "2023-09-25 14:59:36 PDT"
```

Access

- Access the calibration objects with butler.get, providing the RUN collection name

Ex: dark

```
calib = butler.get('dark', instrument='LATISS', detector=0, collections=‘LATISS/calib/DM-43022/refactorCalibs/darkGen.20240227b/20240227T234127Z’)
```

Ex: flat

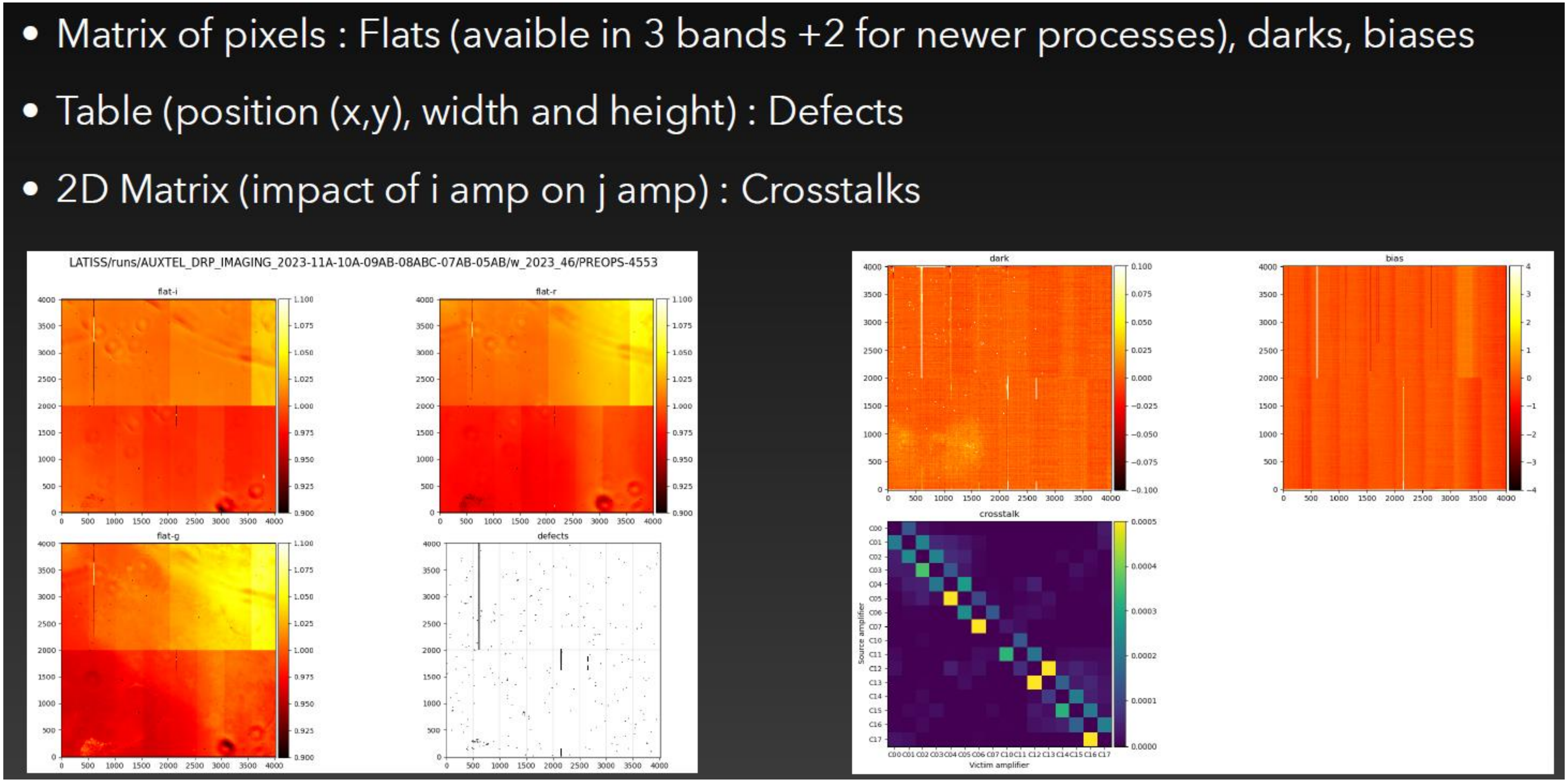
Add the field ‘filter’:

```
calib = butler.get('flat', instrument='LATISS', physical_filter='SDSSg_65mm~empty', detector=0, collections='LATISS/calib/DM-38946/noRGseq/flatGen-g.20230501a/20230501T203920Z‘)
```

- Get arrays

```
arr = calib.getImage().getArray()
arr_var = calib.getVariance().getArray()
```

Visualization



- Matrix of pixels : Flats (avaible in 3 bands +2 for newer processes), darks, biases
- Table (position (x,y), width and height) : Defects
- 2D Matrix (impact of i amp on j amp) : Crosstalks

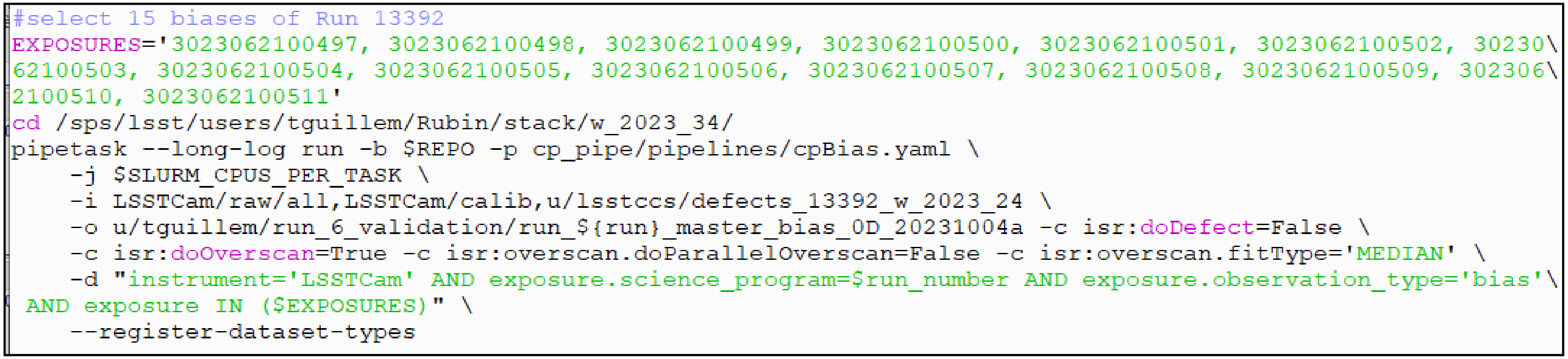
From Nathan (+Rance for defects)

Production

- Guide page: https://pipelines.lsst.io/v/daily/modules/lsst.cp.pipe/constructing-calibrations.html

- Example: master bias production

1) Identify a set of exposures to use as inputs from the repository
2) Run the bias pipeline on these exposures. This pipeline is simple, with a short instrument signal removal (ISR) step that only applies overscan correction and assembles the exposures, before passing them to a combine step that finds the clipped per-pixel mean for the output bias



```
#select 15 biases of Run 13392
EXPOSURES='3023062100497, 3023062100498, 3023062100499, 3023062100500, 3023062100501, 3023062100502, 30230\
62100503, 3023062100504, 3023062100505, 3023062100506, 3023062100507, 3023062100508, 3023062100509, 302306\
2100510, 3023062100511'
cd /sps/lsst/users/tguillem/Rubin/stack/w_2023_34/
pipetask --long-log run -b $REPO -p cp_pipe/pipelines/cpBias.yaml \
    -j $SLURM_CPUS_PER_TASK \
    -i LSSTCam/raw/all,LSSTCam/calib,u/lsstccs/defects_13392_w_2023_24 \
    -o u/tguillem/run_6_validation/run_${run}_master_bias_0D_20231004a -c isr:doDefect=False \
    -c isr:doOverscan=True -c isr:overscan.doParallelOverscan=False -c isr:overscan.fitType='MEDIAN' \
    -d "instrument='LSSTCam' AND exposure.science_program=$run_number AND exposure.observation_type='bias'\
 AND exposure IN ($EXPOSURES)" \
    --register-dataset-types
```

Apply

- To apply your favorite calibrations, just need to be careful with the configuration of ‘-i’ flag.

pipetask uses the first one…

- Example:

  ```
  pipetask … -i LSSTCam/raw/all,u/tguillem/run_6_validation/run_13391_master_bias_0D_20231004a,LSSTCam/calib
  ```

- You can check in the log that the correct collections are used (and also in the output collections after the processing).