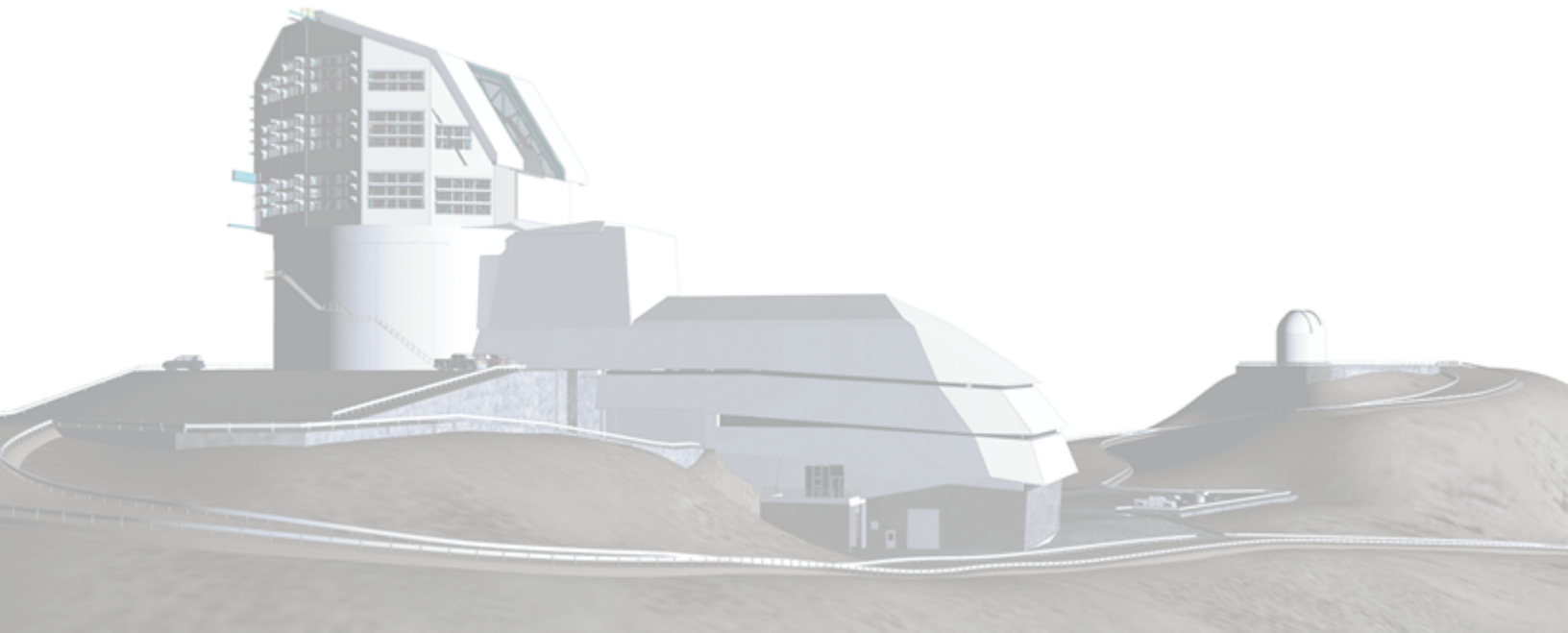


Vera C. Rubin Observatory
Data Management

LVV-P99: Data Management Acceptance Test Campaign 1 Test Plan and Report

Jeffrey Carlin

DMTR-371

Latest Revision: 2023-01-17

Abstract

This is the test plan and report for **Data Management Acceptance Test Campaign 1**, an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2023-01-17 . The most recent change to the document repository was on .

Change Record

Version	Date	Description	Owner name
	2022-07-12	First draft	Jeff Carlin
1.0	2023-01-17	Test campaign LVV-P99 completed and results approved. DM-32340	Jeff Carlin

Document curator: Jeff Carlin
Document source location: `https://github.com/lsst-dm/DMTR-371`
Version from source repository:

Contents

LVV-P99: Data Management Acceptance Test Campaign 1 Test Plan and Report

1 Introduction

1.1 Objectives

The primary goal of this DM acceptance test campaign will be to verify priority 1a DMSR (LSE-61) requirements that have not been verified as part of prior testing and milestones. Any priority 1b, 2, or 3 requirements that have been completed will also be verified.

1.2 System Overview

This test campaign is intended to verify that the DM system satisfies at least half of the priority 1a requirements outlined in the Data Management System Requirements (DMSR; LSE-61), ensuring that we are progressing toward readiness for the installation and operation of LSSTCam. Additional DMSR requirements will be verified in later Acceptance Test Campaigns.

Applicable Documents:
LSE-61: Data Management System (DMS) Requirements
LDM-503 Data Management Test Plan
LDM-639: Data Management Acceptance Test Specification

Tests in this campaign will use data products and artifacts from Data Preview 0.2, which consists of DESC Data Challenge 2 (DC2) simulated data reprocessed using the LSST Science Pipelines. Additional on-sky data from auxTel imaging campaigns will be used when appropriate.

1.3 Document Overview

This document was generated from Jira, obtaining the relevant information from the LVV-P99 Jira Test Plan and related Test Cycles (LVV-C208).

Section 1 provides an overview of the test campaign, the system under test (Acceptance), the applicable documentation, and explains how this document is organized. Section 2 provides additional information about the test plan, like for example the configuration used for this test or related documentation. Section 3 describes the necessary roles and lists the individuals assigned to them.

Section 4 provides a summary of the test results, including an overview in Table 3, an overall assessment statement and suggestions for possible improvements. Section 5 provides detailed results for each step in each test case.

The current status of test plan LVV-P99 in Jira is **Completed** .

1.4 References

[1] **[DMTN-140]**, Comoretto, G., 2021, *Documentation Automation for the Verification and Validation of Rubin Observatory Software*, DMTN-140, URL https://dmtn-140.lsst.io/, Vera C. Rubin Observatory Data Management Technical Note

[2] **[DMTN-178]**, Comoretto, G., 2021, *Docsteady Usecases for Rubin Observatory Constructions*, DMTN-178, URL https://dmtn-178.lsst.io/, Vera C. Rubin Observatory Data Management Technical Note

[3] **[LSE-61]**, Dubois-Felsmann, G., Jenness, T., 2019, *Data Management System (DMS) Requirements*, LSE-61, URL https://lse-61.lsst.io/, Vera C. Rubin Observatory

[4] **[LDM-639]**, Guy, L., Wood-Vasey, W., Bellm, E., et al., 2022, *LSST Data Management Acceptance Test Specification*, LDM-639, URL https://ldm-639.lsst.io/, Vera C. Rubin Observatory Data Management Controlled Document

[5] **[LDM-503]**, O'Mullane, W., Swinbank, J., Juric, M., et al., 2022, *Data Management Test Plan*, LDM-503, URL https://ldm-503.lsst.io/, Vera C. Rubin Observatory Data Management Controlled Document

[6] **[LSE-160]**, Selvy, B., 2013, *Verification and Validation Process*, LSE-160, URL https://ls.st/LSE-160

2 Test Plan Details

2.1 Data Collection

Observing is not required for this test campaign.

2.2 Verification Environment

Most testing will be performed using the Rubin Science Platform (RSP), hosted at the IDF, with auxTel-related tests using the development cluster at the USDF. In particular, we will use version w_2022_32 of the Pipelines for most tests; some tests will use more recent weekly builds of the Pipelines.

2.3 Related Documentation

Jira Attachments	
To LVV-C208 results	DP02ProcessingRuns.png
To LVV-C208 results	DOME_GOGLE_LSST_Queues.png
To LVV-C208 results	butlerStat-PREOPS-905.html
To LVV-C208 results	pandaWfStat-PREOPS-905.html
To LVV-C208 results	pandaStat-PREOPS-905.html

All documents provided as attachments in Jira are downloaded to Github and linked here for convenience. However, since they are not properly versioned, they should be considered informal and therefore not be part of the verification baseline.

2.4 PMCS Activity

Primavera milestones related to the test campaign:

- None

3 Personnel

The personnel involved in the test campaign is shown in the following table.

	T. Plan LVV-P99 owner:	**Jeffrey Carlin**	
	T. Cycle LVV-C208 owner:	**Jeffrey Carlin**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T33	Kian-Tat Lim	Jeffrey Carlin	
LVV-T43	Jim Bosch	Jeffrey Carlin	
LVV-T2697	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2698	Jeffrey Carlin	Jeffrey Carlin	
LVV-T136	Colin Slater	Jeffrey Carlin	
LVV-T2692	Jeffrey Carlin	Jeffrey Carlin	
LVV-T32	Kian-Tat Lim	Jeffrey Carlin	
LVV-T1756	Jeffrey Carlin	Jeffrey Carlin	
LVV-T199	Leanne Guy	Jeffrey Carlin	
LVV-T190	Leanne Guy	Jeffrey Carlin	
LVV-T77	Jim Bosch	Jeffrey Carlin	
LVV-T84	Eli Rykoff	Jeffrey Carlin	
LVV-T151	Colin Slater	Jeffrey Carlin	
LVV-T1232	Colin Slater	Jeffrey Carlin	
LVV-T72	Jim Bosch	Jeffrey Carlin	
LVV-T90	Eli Rykoff	Jeffrey Carlin	
LVV-T137	Colin Slater	Jeffrey Carlin	
LVV-T55	Eric Bellm	Jeffrey Carlin	
LVV-T66	Jim Bosch	Jeffrey Carlin	
LVV-T91	Eli Rykoff	Jeffrey Carlin	
LVV-T126	Eric Bellm	Jeffrey Carlin	
LVV-T1946	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1947	Jeffrey Carlin	Jeffrey Carlin	
LVV-T28	Colin Slater	Jeffrey Carlin	
LVV-T78	Kian-Tat Lim	Jeffrey Carlin	
LVV-T42	Jim Bosch	Jeffrey Carlin	
LVV-T141	Leanne Guy	Leanne Guy	
LVV-T140	Leanne Guy	Leanne Guy	
LVV-T129	Eli Rykoff	Jeffrey Carlin	
LVV-T127	Leanne Guy	Jeffrey Carlin	
LVV-T79	Jim Bosch	Jeffrey Carlin	
LVV-T1830	Jeffrey Carlin	Jeffrey Carlin	

LVV-T145	Leanne Guy	Jeffrey Carlin
LVV-T144	Kian-Tat Lim	Jeffrey Carlin
LVV-T74	Eric Bellm	Jeffrey Carlin

4 Test Campaign Overview

4.1 Summary

T. Plan LVV-P99:	**Data Management Acceptance Test Campaign 1**			Completed
T. Cycle LVV-C208:	**Data Management Acceptance Test Campaign 1**			Done
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T33	1	Initial Pass	The python script to execute this test is attached to the Test Report github repository in scripts/test_LVV-T32.py. (This test was executed with LVV-T32.)	
LVV-T43	1	Pass	The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T43.ipynb".	
LVV-T2697	1	Pass	The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T2697.ipynb".	
LVV-T2698	1	Pass	The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T2698.ipynb".	
LVV-T136	1	Pass	The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T136.ipynb".	
LVV-T2692	1	Pass	The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T2692.ipynb".	
LVV-T32	1	Pass	The python script to execute this test is attached to the Test Report github repository in scripts/test_LVV-T32.py.	
LVV-T1756	1	Pass		
LVV-T199	1	Pass		
LVV-T190	1	Pass		
LVV-T77	1	Pass	The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T77.py.	
LVV-T84	1	Pass	Results are in the notebook "test_LVV-T84.ipynb" attached to the Test Report repository.	

LVV-T151	1	Pass	Results of the test execution can be found in the notebook “test_LVV-T151.ipynb” attached to the github repository.
LVV-T1232	1	Pass	
LVV-T72	1	Pass	The executed notebook was saved in the repository associated with this campaign’s test report as “notebooks/test_LVV-T72.ipynb”.
LVV-T90	1	Pass	Results are in the notebook “test_LVV-T90.ipynb” attached to the Test Report repository.
LVV-T137	1	Pass	
LVV-T55	1	Pass	The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T55.py.
LVV-T66	1	Pass	The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T66.py.
LVV-T91	1	Pass	Results are in the notebook “test_LVV-T91.ipynb” attached to the Test Report repository.
LVV-T126	1	Pass	Results are in the notebook “test_LVV-T126.ipynb” attached to the Test Report repository.
LVV-T1946	1	Pass	Executed at the IDF using Science Pipelines version w_2022_32. This test case can be executed by running the script test_LVV-T1946.py, which is available in the test report github repository’s “scripts/” directory. Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.

			Executed at the IDF using Science Pipelines version w_2022_32.
LVV-T1947	1	Pass	This test case can be executed by running the script test_LVV-T1947.py, which is available in the test report github repository's "scripts/" directory. Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.
			Executed at the IDF using Science Pipelines version w_2022_32.
LVV-T28	1	Pass	This test case can be executed by running the script test_LVV-T28.py, which is available in the test report github repository's "scripts/" directory. Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.
LVV-T78	1	Pass	The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T78.py.
LVV-T42	1	Pass	Results are in the notebook "test_LVV-T42.ipynb" attached to the Test Report repository.
LVV-T141	1	Pass	
LVV-T140	1	Pass	

			Executed at the IDF using Science Pipelines version w_2022_32. A portion of this test case can be executed by running the script test_LVV-T129.py, which is available in the test report github repository's "scripts/" directory. Qserv tables (i.e., Object tables) were checked via the notebook test_LVV-T129.ipynb, which is available in the test report github repository's "notebooks" directory.
LVV-T129	1	Initial Pass	This Test Case is granted status of "Initial Pass" because the DIAObject table in Qserv does not clearly demonstrate that fluxes are provided in nJy. Though the fluxes are shown to be reasonable values if they are in nJy (and inspection of code could confirm this), we choose not to grant this test a full PASS until the DIAObject table is clearly reporting fluxes in nJy.
LVV-T127	1	Pass	See attached artifact notebook "test_LVV-T127.ipynb" for details.
LVV-T79	1	Pass	
LVV-T1830	1	Pass	
LVV-T145	1	Pass	
LVV-T144	1	Pass	
LVV-T74	1	Pass	See attached artifact notebook "test_LVV-T74.ipynb" for details.

Table 3: Test Campaign Summary

4.2 Overall Assessment

In this test campaign, we have successfully verified 28 unique requirements from LSE-61 via the execution of 35 Test Cases (of which 34 passed, and one was deemed an initial pass). Of these requirements, 15 are of priority 1a, 11 are priority 1b, and 2 are priority 2. The set of requirements tested in this campaign cover many aspects of the DM system, including software pipelines, databases, the Science Platform, and facilities, with the majority related to the Science Pipelines. Many of these tests were executed using the DP0.2 dataset at the Interim Data Facility (IDF), with a handful performed at the US Data Facility (USDF) using Auxtel

data. The results of this campaign give a picture of many of the foundational capabilities required for DM to successfully gather, process, and distribute LSST data. The majority of remaining requirements to be verified concern more detailed or complicated capabilities.

4.3 Recommended Improvements

Some difficulty arose because we executed this test campaign during the transition from the previous data facility to the current USDF. Thankfully, many requirements can be verified using the simulated DP0.2 data set on the IDF. We recommend that future test campaigns should not be initiated until the required facilities, software, and hardware are confirmed to be available. Another important consideration for future verification is the variety of activities required to verify the DM system, which can make testing time consuming and difficult. As many capabilities as possible should be tested in an automated way during (re-)processing campaigns using, for example, the 'faro' or 'analysis_tools' Science Pipelines packages.

5 Detailed Test Results

5.1 Test Cycle LVV-C208

Open test cycle *Data Management Acceptance Test Campaign 1* in Jira.

Test Cycle name: Data Management Acceptance Test Campaign 1
Status: Done

This test cycle verifies a subset of DMSR (LSE-61) requirements in order to verify their completion and readiness for LSST Operations (i.e., that the requirements laid out in LSE-61 have been met by the DM Systems). Testing will use data products and artifacts from Data Preview 0.2 reprocessing of DESC DC2 data.

5.1.1 Software Version/Baseline

Using Science Pipelines version w_2022_32 on the interim data facility (IDF) at data.lsst.cloud.

5.1.2 Configuration

Not provided.

5.1.3 Test Cases in LVV-C208 Test Cycle

5.1.3.1 LVV-T33 - Verify implementation of Raw Science Image Metadata

Version **1**. Status **Approved**. Open *LVV-T33* test case in Jira.

Verify successful ingestion of raw data and that image metadata is present and queryable.

Preconditions:

Execution status: **Initial Pass**

Final comment:
The python script to execute this test is attached to the Test Report github repository in scripts/test_LVV-T32.py. (This test was executed with LVV-T32.)

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify (or gather) a dataset of raw science images.

Expected Result

Actual Result
For this test, we use weekly Science Pipelines version 'w_2022_45' at the USDF, examining recently gathered AuxTel imaging data from the campaign detailed in SITCOM-505. The data were obtained in late October 2022, and processed at USDF with pipelines version 'w_2022_44'.

In particular, these data are at '/repo/embargo', in collection 'u/huanlin/auxtel_oga_panda_test_2022102528_w44'.

In the attached script, we instantiate a butler pointing to that repo/collection:

```
from lsst.daf.butler import Butler

repo = '/repo/embargo'
collection = 'u/huanlin/auxtel_oga_panda_test_2022102528_w44'
butler = Butler(repo, collections=collection)
```

The attached script retrieves the selected raw image via the following:

```
dataId = {'exposure':2022092900894, 'detector':0}
raw = butler.get('raw', dataId=dataId)
md = raw.getMetadata()
```

Step 2 Step Execution Status: **Pass**

Description

Verify that time of exposure start/end, site metadata, telescope metadata, and camera metadata are stored in DMS system.

Expected Result

Raw image data contain the required metadata.

Actual Result

The attached script also extracts the metadata attached to the image. The following code extracts the metadata and prints each entry to the screen:

```
md_dict = md.toDict()
for item in md_dict.items():
print(item)
```

The following results were printed to the screen:
Metadata:
('SIMPLE', True)
('EXTEND', True)
('CCD_MANU', 'ITL')
('CCD_TYPE', '3800C')
('BINX', 1)
('BINY', 1)
('CCDGAIN', 1.0)
('CCDNOISE', 10.0)
('CCDSLOT', 'S00')
('RAFTBAY', 'R00')
('FIRMWARE', '11384004')
('PLATFORM', 'auxtel')
('CONTNUM', '189216ee')
('DAQVERS', 'R5-V0.6 2021-10-06T19:50:32Z (1800122)')
('DAQPART', 'lat')
('DAQFOLD', 'raw')
('SEQFILE', 'FP_ITL_2s_ir2_v25.seq')
('SEQNAME', 'FP_ITL_2s_ir2_v25.seq')

('SEQCKSUM', '2552520002')
('LSST_NUM', 'ITL-3800C-068')
('CCD_SERN', '20862')
('REBNAME', 'Unknown')
('RAFTNAME', 'AuxTel-Raft')
('FPVERS', '1.1.4')
('IHVERS', '1.0.30')
('STUTTER ROWS', 0)
('STUTTER DELAY', 0.0)
('STUTTER NSHIFTS', 0)
('FILTPOS', None)
('CCDTEMP', -90.7174)
('COMMENT', ['', '—- Date, night and basic image information —-', '', '—- Telescope info, location, observer —-', '', '—- info, etc. —-', '', '—- TAN Projection —-', '', '—- Image-identifying used to build OBS-ID —-', '', '—- Additional Keys Information from Camera —-', '', '—- Image sequence numbers —-', '', '—- Test Stand information —-', '', '—- Information from Camera (Common block) —-', '', '—- Information from Camera —-', '', '—- Filter/grating information —-', '', '—- Exposure-related information —-', '', '—- Weather information —-', '', '—- Header information —-', '', '—- Hierarch information for CSC Simulatiom Mode —-', '—- Checksums —-', '—- Checksums —-'])
('DATE', '2022-09-30T08:19:19.187')
('MJD', 59852.34674984962)
('IMGTYPE', 'OBJECT')
('DATE-OBS', '2022-09-30T08:18:28.952')
('MJD-OBS', 59852.346168425865)
('DATE-TRG', '2022-09-30T08:18:42.184')
('MJD-TRG', 59852.34632157395)
('OBSID', 'AT_O_20220929_000894')
('DATE-BEG', '2022-09-30T08:18:28.952')
('MJD-BEG', 59852.346168425865)
('DATE-END', '2022-09-30T08:19:19.187')
('MJD-END', 59852.34674984962)
('GROUPID', '2022-09-30T08:10:20.478')
('BUNIT', 'adu')
('TIMESYS', 'TAI')
('INSTRUME', 'LATISS')
('TELESCOP', 'LSST AuxTelescope')
('OBS-LONG', -70.749417)
('OBS-LAT', -30.244639)
('OBS-ELEV', 2663.0)
('OBSGEO-X', 1818938.94)
('OBSGEO-Y', -5208470.95)
('OBSGEO-Z', -3195172.08)
('FACILITY', 'Vera C. Rubin Observatory')
('RA', 48.41552041666668)
('DEC', -30.22323527777778)

('RASTART', 48.41613010222692)
('DECSTART', -30.209342398754412)
('RAEND', 48.41616329891139)
('DECEND', -30.20934820761029)
('ROTPA', 359.9987453403358)
('ROTCOORD', 'sky')
('HASTART', 0.9417104442537305)
('ELSTART', 77.89912050802377)
('AZSTART', -92.99574590867269)
('AMSTART', 1.0227214941767613)
('AHAEND', None)
('AELEND', None)
('AAZEND', None)
('AAMEND', None)
('TRACKSYS', None)
('FOCUSZ', 0.006039788480848074)
('OBJECT', 'HD 20187')
('INSTPORT', 2)
('ATM3PORT', 2)
('DOMEAZ', 267.67)
('SHUTLOWR', 0.0)
('SHUTUPPR', 100.0)
('TESTTYPE', 'OBJECT')
('CAMCODE', 'AT')
('CONTRLLR', 'O')
('DAYOBS', '20220929')
('SEQNUM', 894)
('PROGRAM', 'SITCOM-479')
('REASON', 'PSF+Photometry_Validation')
('CURINDEX', 1)
('MAXINDEX', 1)
('TSTAND', None)
('IMAGETAG', '493e97dca7a3ec0a')
('OBSANNOT', '')
('TEMP_SET', -94.15)
('GRATING', 'empty_1')
('GRATBAND', 'EMPTY')
('GRATSLOT', 0)
('LINSPOS', 66.99800109863281)
('FILTBAND', 'r')
('FILTER', 'SDSSr')
('FILTSLOT', 1)
('EXPTIME', 50.0)
('DARKTIME', 50.2347)

('SHUTTIME', None)
('AIRTEMP', None)
('PRESSURE', None)
('HUMIDITY', None)
('WINDSPD', None)
('WINDDIR', None)
('SEEING', 0.9242383241653442)
('FILENAME', 'AT_O_20220929_000894_R00_S00.fits')
('HEADVER', 2)
('SIMULATE ATMCS', None)
('SIMULATE ATHEXAPOD', 0)
('SIMULATE ATPNEUMATICS', None)
('SIMULATE ATDOME', 0)
('SIMULATE ATSPECTROGRAPH', 0)
('XTENSION', 'BINTABLE')
('BITPIX', 8)
('NAXIS', 2)
('NAXIS1', 8)
('NAXIS2', 2048)
('PCOUNT', 929978)
('GCOUNT', 1)
('TFIELDS', 1)
('TTYPE1', 'COMPRESSED_DATA')
('TFORM1', '1PB(523)')
('ZIMAGE', True)
('ZTILE1', 576)
('ZTILE2', 1)
('ZCMPTYPE', 'RICE_1')
('ZNAME1', 'BLOCKSIZE')
('ZVAL1', 32)
('ZNAME2', 'BYTEPIX')
('ZVAL2', 4)
('ZTENSION', 'IMAGE')
('ZBITPIX', 32)
('ZNAXIS', 2)
('ZNAXIS1', 576)
('ZNAXIS2', 2048)
('ZPCOUNT', 0)
('ZGCOUNT', 1)
('ZHECKSUM', '5GGXAF9V7FEVAF9V')
('CHANNEL', 1)
('EXTNAME', 'Segment10')
('CCDSUM', '1 1')
('DATASEC', '[4:512,1:2000]')

('DETSEC', '[509:1,1:2000]')
('DETSIZE', '[1:4072,1:4000]')
('DTV1', 513), ('DTV2', 0)
('DTM1_1', -1.0), ('DTM2_2', 1.0), ('DTM1_2', 0.0), ('DTM2_1', 0.0)
('CTYPE1A', 'Seg_X'), ('CTYPE2A', 'Seg_Y')
('CRPIX1A', 0.0), ('CRPIX2A', 0.0)
('CRVAL1A', 2001.0), ('CRVAL2A', 513.0)
('BSCALE', 1.0)
('BZERO', 0.0)
('INHERIT', True)
('ZDATASUM', '261725759')
('CHECKSUM', 'JiD1JiA0JiA0JiA0')
('DATASUM', '546670105')
('TEMP1', 14.6875), ('TEMP2', 15.75), ('TEMP3', 14.5625)
('TEMP4', 13.8125), ('TEMP5', 13.375), ('TEMP6', 13.0)
('ATEMPU', 4.82143), ('ATEMPL', 4.79911)
('RTDTEMP', 319.806)
('DIGPS_V', 5.25), ('DIGPS_I', 694.0)
('ANAPS_V', 7.2), ('ANAPS_I', 304.25)
('CLKHPS_V', 12.025), ('CLKHPS_I', 31.25), ('CLKLPS_V', -11.9846)
('ODPS_V', 32.05), ('ODPS_I', 30.0)
('HTRPS_V', 0.0), ('HTRPS_W', 0.0)
('PCKU_V', 1.6977), ('PCKL_V', -8.29127)
('SCKU_V', 5.15739), ('SCKL_V', -5.10488)
('RGU_V', 8.28484), ('RGL_V', -2.07496)
('ODV', 32.05), ('ODI', 28.2), ('GDP', 20.0), ('RDP', 13.0), ('OGP', -2.0), ('ODP', 25.0)
('CSGATEP', 1.0)
('SCK_LOWP', -5.0), ('SCK_HIP', 5.0)
('PCK_LOWP', -8.0), ('PCK_HIP', 2.0)
('RG_LOWP', -2.0), ('RG_HIP', 8.0)
('AP0_RC', 3), ('AP1_RC', 3)
('AP0_GAIN', 0), ('AP1_GAIN', 0)
('AP0_CLMP', 0), ('AP1_CLMP', 0)
('AP0_AF1', 0), ('AP1_AF1', 0)
('AP0_TM', 0), ('AP1_TM', 0)
('HVBIAS', 'ON')
('ANAV', 7.2), ('ANAI', 306.9)
('DIGV', 5.25), ('DIGI', 674.7)
('FANV', 12.0), ('FANI', 104.6)
('CLKHIGHV', 12.0), ('CLKHIGHI', 30.4), ('CLKLOWV', 12.0), ('CLKLOWI', 30.8)
('HVBIASV', -50.0), ('HVBIASI', -5.24095e-12)
('DPHIV', 10.0), ('DPHII', 6.4)
('OTMV', 5.0), ('OTMI', 460.9)
('AUXV', 0.0), ('AUXI', 0.0)

('FPDAQCS', '4029549637'), ('FPHIDCS', '3666005214'), ('FPINSCS', '969329422')
('FPLIMCS', '840592320'), ('FPRTCCS', '2654667822'), ('FPRTCSCS', '1965734659')
('FPRCS', '1524324489'), ('FPRLCS', '143804913'), ('FPRPCS', '2223968974')
('FPSEQCS', '3259110827'), ('FPTIMCS', '612664540')
('RPLIMCS', '302651546'), ('RPPOWCS', '3969790867'), ('RPTIMCS', '81899862')
('HIERARCH ASTRO METADATA FIX MODIFIED', True)
('HIERARCH ASTRO METADATA FIX DATE', '2022-11-08T16:17:22.029574')
('HIERARCH ASTRO METADATA FIX VERSION', 'g1625dc41fd+0703681425')

By examination of this metadata, we determine that the correct information has been included. This includes the start/end times of the exposure (referenced to TAI), site information including observatory location, seeing, etc., telescope metadata regarding its pointing, sensor readings, and camera and program metadata.

5.1.3.2 LVV-T43 - Verify implementation of Background Model Calculation

Version **1**. Status **Approved**. Open *LVV-T43* test case in Jira.

Verify that Processed Visit Images produced by the DRP and AP pipelines have had a model of the background subtracted, and that this model is persisted in a way that permits the background subtracted from any CCD to be retrieved along with the image for that CCD.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T43.ipynb".

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description
Identify a dataset with processed visit images in multiple filters.

Expected Result

Actual Result
For this test we use the 'rc2_subset' dataset, which is a small collection of HSC data often used for testing. In particular, we use a version processed through all DRP pipeline steps on the USDF with Pipelines version w_2022_32.

Step 2 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result
Butler repo available for reading.

Actual Result
from lsst.daf.butler import Butler

repo = '/sdf/group/rubin/u/jcarlin/repos/rc2_subset/SMALL_HSC'
coll = 'u/jcarlin/step4'
# Note: the ccdVisitTable resides in a separate collection because it was created after all processing was done.
table_coll = 'u/jcarlin/visit_table'
butler = Butler(repo, collections=[coll, table_coll])

Step 3 Step Execution Status: **Pass**

Description
Display an image of the background model for a full CCD. Repeat this for all available filters, and confirm that the background is smoothly varying and defined over the full CCD.

Expected Result
Well-formed background covering the entire CCD for all CCDs in all filters.

Actual Result
CCD visit/detector combinations were selected at random and the corresponding dataIds (datarefs) created. To extract the background, the following line was executed for each dataId:

bb = butler.get('calexpBackground', dataId = data_ids[ii])

These were displayed alongside the final calexp image.

Additionally, a larger subset of dataIds was selected, from which we test whether (a) *all* calexps have an associated background model, and (b) the background model is well-formed and populated with finite values for all pixels. The result of this test, seen in the test notebook, is as follows:

```
All CCDs have an associated background.
```

If any of the 100 randomly selected CCDs had a malformed (or non-existent) background model, this statement would not return True.

The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T43.ipynb".

Step 4 Step Execution Status: **Pass**

Description
Confirm that the pixel values of the calexp + calexpBackground are approximately equal to those of the postISR-CCD image.

Expected Result
All calexp+calexpBackground images should have pixel values *approximately* equal to those of postISRCCD images. Small differences are expected due to cosmic-ray repair and other similar corrections, but the median should be equal.

Actual Result
In the attached notebook, we looped over the dataIds, selecting the original postISRCCD image, the calexp, and the background for each one. We calculated the median value of postISRCCD - (calexp + background), and demonstrated via an assert statement that this value is zero for all images.

5.1.3.3 LVV-T2697 - Verify implementation of Catalog Data Product Access

Version **1**. Status **Approved**. Open *LVV-T2697* test case in Jira.

Verify that available catalog data products can be listed and retrieved.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T2697.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working in a notebook entitled “test_LVV-T2697.ipynb” on the Interim Data Facility at data.lsst.cloud:

from lsst.daf.butler import Butler

config = ‘dp02’
collection = ‘2.2i/runs/DP0.2’
butler = Butler(config, collections=collection)

Step 2 Step Execution Status: **Pass**

Description

Execute a Butler query to identify and list ‘src’ catalogs (measured from ‘calexp’ images).

Expected Result

A list of DataIds identifying ‘src‘ catalogs.

Actual Result

First, select a tract, patch, and detector to query. (These are known a priori to exist in the DP0.2 data.)

tract = 3828
patch = 42
detector = 42

Query the Butler registry as follows:

data_refs = butler.registry.queryDatasets(
datasetType="src",
where=f"tract={tract} and patch={patch} and detector={detector} and skymap='DC2'")

data_ids = []

for data_ref in data_refs:
print(data_ref.dataId.full)
data_ids.append(data_ref.dataId.full)

This prints the following list of dataIds to the screen:

```
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 414872}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 457676}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 713247}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 1013706}
{band: 'y', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'y_sim_1.4', visit_system: 1, visit: 1138156}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1155522}
{band: 'z', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'z_sim_1.4', visit_system: 1, visit: 1174350}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1192139}
```

We have thus demonstrated that catalog data products can be queried and listed via the Butler.

Step 3 Step Execution Status: **Pass**

Description

Using 'Butler.get()', retrieve a 'src' catalog using one of the DataIds returned in the previous step. Print a table version of this catalog, then its schema, and confirm that the catalog is well-formed.

Expected Result

Confirmation that a 'src' catalog has been retrieved by confirming that is has measurements of positions, fluxes, and shapes, and contains flags describing measurement quality.

Actual Result

Retrieve a src catalog via the following line:

---

src = butler.get('src', dataId=data_ids[0])

Print the catalog contents to the screen:
src.asAstropy()

(For brevity, the output from this is not shown in this panel. See the attached notebook for the output information.)

Print and examine the catalog schema to confirm that it contains expected measurements. A portion of the output is as follows:

```
The catalog has  265  columns.

 Catalog schema:
```

[7]:

```
Schema(
    (Field['L'](name="id", doc="unique ID"), Key<L>(offset=0, nElements=1)),
    (Field['Angle'](name="coord_ra", doc="position in ra/dec"), Key<Angle>(offset=8, nElements=1)),
    (Field['Angle'](name="coord_dec", doc="position in ra/dec"), Key<Angle>(offset=16, nElements=1)),
    (Field['L'](name="parent", doc="unique ID of parent source"), Key<L>(offset=24, nElements=1)),
   (Field['Flag'](name="calib_detected", doc="Source was detected as an icSource"), Key['Flag'](offset=32, bit=0)),
   (Field['Flag'](name="calib_psf_candidate", doc="Flag set if the source was a candidate for PSF determination, as determine
   (Field['Flag'](name="calib_psf_used", doc="Flag set if the source was actually used for PSF determination, as determined by
   (Field['Flag'](name="calib_psf_reserved", doc="set if source was reserved from PSF determination"), Key['Flag'](offset=32
   (Field['I'](name="deblend_nChild", doc="Number of children this object has (defaults to 0)"), Key<I>(offset=40, nElements=
   (Field['Flag'](name="deblend_deblendedAsPsf", doc="Deblender thought this source looked like a PSF"), Key['Flag'](offset=
   (Field['D'](name="deblend_psfCenter_x", doc="If deblended-as-psf, the PSF centroid", units="pixel"), Key<D>(offset=48, nE
   (Field['D'](name="deblend_psfCenter_y", doc="If deblended-as-psf, the PSF centroid", units="pixel"), Key<D>(offset=56, nE
   (Field['D'](name="deblend_psf_instFlux", doc="If deblended-as-psf, the instrumental PSF flux", units="count"), Key<D>(off
...

   (Field['D'](name="base_CircularApertureFlux_3_0_instFlux", doc="instFlux within 3.000000-pixel aperture", units="count")
```

```
(Field['D'](name="base_CircularApertureFlux_3_0_instFluxErr", doc="1-sigma instFlux uncertainty", units="count"), Key<D>
(Field['Flag'](name="base_CircularApertureFlux_3_0_flag", doc="General Failure Flag"), Key['Flag'](offset=32, bit=59)),
(Field['Flag'](name="base_CircularApertureFlux_3_0_flag_apertureTruncated", doc="aperture did not fit within measurement
(Field['Flag'](name="base_CircularApertureFlux_3_0_flag_sincCoeffsTruncated", doc="full sinc coefficient image did not f
(Field['D'](name="base_CircularApertureFlux_4_5_instFlux", doc="instFlux within 4.500000-pixel aperture", units="count")
```

...

We have confirmed that catalog data products can be accessed via the Butler using the query results from the previous step.

Step 4 Step Execution Status: **Pass**

Description

Repeat steps 2 and 3 for catalogs derived from (a) difference images, and (b) coadd images.

Expected Result

Actual Result

See the attached notebook, wherein we repeated the above steps for 'goodSeeingDiff_diaSrc' and 'deepCoadd_forced_src' catalogs.

5.1.3.4 LVV-T2698 - Verify implementation of Catalog Metadata Access

Version **1**. Status **Approved**. Open *LVV-T2698* test case in Jira.

Verify that available catalog data products' metadata can be listed and retrieved.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test re-

---

port as “notebooks/test_LVV-T2698.ipynb”.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working in a notebook entitled “test_LVV-T2692.ipynb” on the Interim Data Facility at data.lsst.cloud:

from lsst.daf.butler import Butler

config = ‘dp02’
collection = ‘2.2i/runs/DP0.2’
butler = Butler(config, collections=collection)

Step 2 Step Execution Status: **Pass**

Description

Execute a Butler query to identify and list ‘src‘ catalogs (measured from ‘calexp‘ images).

Expected Result

A list of DataIds identifying ‘src‘ catalogs.

Actual Result

First, select a tract, patch, and detector to query. (These are known a priori to exist in the DP0.2 data.)

tract = 3828
patch = 42
detector = 42

Query the Butler registry as follows:

data_refs = butler.registry.queryDatasets(
datasetType="src",
where=f"tract={tract} and patch={patch} and detector={detector} and skymap='DC2'")

data_ids = []

for data_ref in data_refs:
print(data_ref.dataId.full)
data_ids.append(data_ref.dataId.full)

This prints the following list of dataIds to the screen:

```
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 414872}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 457676}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 713247}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 1013706}
{band: 'y', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'y_sim_1.4', visit_system: 1, visit: 1138156}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1155522}
{band: 'z', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'z_sim_1.4', visit_system: 1, visit: 1174350}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1192139}
```

Step 3 Step Execution Status: **Pass**

Description

Use 'butler.registry.queryDimensionRecords()' to identify the metadata associated with a patch, then a visit, for one of the dataIds returned in the query. Print the results to the screen.

Expected Result

Output printed to the screen should include a list of patch metadata including the associated tract, id, and bounding region, and a list of visit metadata which includes information such as exposure time, date of observation, visit id, filter, etc.

Actual Result

Extract "patch" metadata corresponding to a single data_ref:

calexp_patch_metadata = butler.registry.queryDimensionRecords('patch', dataId=data_ref.dataId.full)

Printing the first of these results to the screen yields:

```
patch:
  skymap: 'DC2'
  tract: 4026
  id: 0
  cell_x: 0
  cell_y: 0
 region: ConvexPolygon([UnitVector3d(0.43026204775397675, 0.6883194991736487, -0.5840298257108897), UnitVector3d(0.426896
```

Repeat, but now for "visit" information:
calexp_visit_metadata = butler.registry.queryDimensionRecords('visit', dataId=data_ref.dataId.full)

```
visit:
  instrument: 'LSSTCam-imSim'
  id: 1192139
  physical_filter: 'r_sim_1.4'
  visit_system: 1
```

```
  name: '1192139'
  day_obs: 20261013
  exposure_time: 30.0
  target_name: 'UNKNOWN'
  observation_reason: 'imsim'
  science_program: '1192139'
  zenith_angle: 33.24096737781019
 region: ConvexPolygon([UnitVector3d(0.4540277748455429, 0.6654318675118485, -0.5925024973521197), UnitVector3d(0.4436108
 timespan: Timespan(begin=astropy.time.Time('2026-10-14 04:22:46.411200', scale='tai', format='iso'), end=astropy.time.Tim
```

Finally, repeat once more for “detector” info:

calexp_detector_metadata = butler.registry.queryDimensionRecords(‘detector’, dataId=data_ref.dataId.full)

```
detector:
  instrument: 'LSSTCam-imSim'
  id: 42
  full_name: 'R11_S20'
  name_in_raft: 'S20'
  raft: 'R11'
  purpose: 'SCIENCE'
```

We have thus demonstrated the ability to list and access metadata via the Butler.

Step 4 Step Execution Status: **Pass**

Description

Use ‘butler.get()’ to directly retrieve a dataset with “metadata” in its DatasetType name (e.g., ‘calibrate_metadata’). Print the returned metadata values to the screen.

Expected Result

Detailed information about the processing executed during the ‘calibrate’ task is printed to the screen.

Actual Result

To demonstrate a method of accessing a different type of metadata, execute the following:

---

calibrate_metadata = butler.get('calibrate_metadata', dataId = data_ref.dataId.full)

for v in calibrate_metadata.values():
print(v)

```
runStartUtc = "2022-01-03T14:27:01.303472"
runStartCpuTime = 10576.362046784
runStartUserTime = 10193.241786000
runStartSystemTime = 383.12026400000
runStartMaxResidentSetSize = 3084460
runStartMinorPageFaults = 99609347
runStartMajorPageFaults = 8
runStartBlockInputs = 63528
runStartBlockOutputs = 78739800
runStartVoluntaryContextSwitches = 2237238
runStartInvoluntaryContextSwitches = 8650111
runEndUtc = "2022-01-03T14:27:30.092458"
runEndCpuTime = 10605.141212355
...
```

```
fitWcsEndUserTime = [ 10222.800983000, 10223.033200000, 10223.264442000 ]
fitWcsEndSystemTime = [ 383.30272600000, 383.41161300000, 383.52557000000 ]
fitWcsEndMaxResidentSetSize = [ 3084460, 3084460, 3084460 ]
fitWcsEndMinorPageFaults = [ 99633457, 99633457, 99633457 ]
fitWcsEndMajorPageFaults = [ 8, 8, 8 ]
fitWcsEndBlockInputs = [ 63528, 63528, 63528 ]
fitWcsEndBlockOutputs = [ 78752760, 78752760, 78752768 ]
fitWcsEndVoluntaryContextSwitches = [ 2237936, 2237938, 2237940 ]
fitWcsEndInvoluntaryContextSwitches = [ 8650272, 8650356, 8650391 ]
```

We show a portion of the results that are returned via 'butler.get()'. The full output is included in the attached notebook.

This test execution has demonstrated two ways to list and access catalog metadata via the Butler.

5.1.3.5 LVV-T136 - Verify implementation of Image Data Product Access

Version **1**. Status **Approved**. Open *LVV-T136* test case in Jira.

Verify that available image data products can be listed and retrieved.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T136.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result
Butler repo available for reading.

Actual Result
Working in a notebook entitled "test_LVV-T136.ipynb" on the Interim Data Facility at data.lsst.cloud:

```
from lsst.daf.butler import Butler


config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)
```

Step 2 Step Execution Status: **Pass**

Description

Execute a Butler query to identify and list 'calexp' images.

Expected Result

A list of DataIds identifying 'calexp' images.

Actual Result

First, select a tract, patch, and detector to query. (These are known a priori to exist in the DP0.2 data.)

```
tract = 3828
patch = 42
detector = 42
```

Query the Butler registry as follows:

```
data_refs = butler.registry.queryDatasets(
datasetType="calexp",
where=f"tract={tract} and patch={patch} and detector={detector} and skymap='DC2'")


data_ids = []


for data_ref in data_refs:
print(data_ref.dataId.full)
data_ids.append(data_ref.dataId.full)
```

This prints the following list of dataIds to the screen:

```
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 414872}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 457676}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 713247}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 1013706}
{band: 'y', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'y_sim_1.4', visit_system: 1, visit: 1138156}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1155522}
{band: 'z', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'z_sim_1.4', visit_system: 1, visit: 1174350}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1192139}
```

We have thus demonstrated that image data products can be queried and listed via the Butler.

Step 3 Step Execution Status: **Pass**

Description

Using ‘Butler.get()‘, retrieve a ‘calexp‘ image using one of the DataIds returned in the previous step. Display this image and confirm that it is well-formed.

Expected Result

Confirmation that a ‘calexp‘ image has been retrieved by displaying the image and examining some statistics of the pixel values.

Actual Result

Retrieve an image via the following line:
calexp = butler.get(‘calexp’, dataId=data_ids[0])

In the attached notebook, one can see the displayed image. We also checked statistics of its pixel values to confirm that it is well-formed:

print(’ image median stddev (of image pixel values)’)
print(’calexp: ’, np.nanmedian(calexp.image.array), np.nanstd(calexp.image.array))
print(’variance: ’, np.nanmedian(calexp.variance.array), np.nanstd(calexp.variance.array))

```
 image           median     stddev    (of image pixel values)
```

```
calexp:             0.65257454 307.72455
variance:           1704.5837 449.57498
```

We have confirmed that image data products can be accessed via the Butler using the query results from the previous step.

Step 4 Step Execution Status: **Pass**

Description

Repeat steps 2 and 3 for (a) difference images, and (b) coadd images.

Expected Result

Actual Result

See the attached notebook, wherein we repeated the above steps for 'goodSeeingDiff_differenceExp' and 'deepCoadd_calexp' images.

5.1.3.6 LVV-T2692 - Verify implementation of Image Metadata Access

Version **1**. Status **Approved**. Open *LVV-T2692* test case in Jira.

Verify that available image data products' metadata can be listed and retrieved.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T2692.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working in a notebook entitled "test_LVV-T2692.ipynb" on the Interim Data Facility at data.lsst.cloud:

from lsst.daf.butler import Butler

config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)

Step 2 Step Execution Status: **Pass**

Description

Execute a Butler query to identify and list 'calexp' images.

Expected Result

A list of DataIds identifying 'calexp' images.

Actual Result

First, select a tract, patch, and detector to query. (These are known a priori to exist in the DP0.2 data.)

tract = 3828
patch = 42
detector = 42

Query the Butler registry as follows:

data_refs = butler.registry.queryDatasets(
datasetType="calexp",
where=f"tract={tract} and patch={patch} and detector={detector} and skymap='DC2'")

data_ids = []

for data_ref in data_refs:
print(data_ref.dataId.full)
data_ids.append(data_ref.dataId.full)

This prints the following list of dataIds to the screen:

```
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 414872}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 457676}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 713247}
{band: 'i', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'i_sim_1.4', visit_system: 1, visit: 1013706}
{band: 'y', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'y_sim_1.4', visit_system: 1, visit: 1138156}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1155522}
{band: 'z', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'z_sim_1.4', visit_system: 1, visit: 1174350}
{band: 'r', instrument: 'LSSTCam-imSim', detector: 42, physical_filter: 'r_sim_1.4', visit_system: 1, visit: 1192139}
```

Step 3 Step Execution Status: **Pass**

Description

Use 'butler.registry.queryDimensionRecords()' to identify the metadata associated with a patch, then a visit, for one of the dataIds returned in the query. Print the results to the screen.

Expected Result

Output printed to the screen should include a list of patch metadata including the associated tract, id, and bounding region, and a list of visit metadata which includes information such as exposure time, date of observation, visit id, filter, etc.

Actual Result

Extract “patch” metadata corresponding to a single data_ref:

calexp_patch_metadata = butler.registry.queryDimensionRecords(‘patch’, dataId=data_ref.dataId.full)

Printing the first of these results to the screen yields:

```
patch:
  skymap: 'DC2'
  tract: 4026
  id: 0
  cell_x: 0
  cell_y: 0
 region: ConvexPolygon([UnitVector3d(0.43026204775397675, 0.6883194991736487, -0.5840298257108897), UnitVector3d(0.426896
```

Repeat, but now for “visit” information:
calexp_visit_metadata = butler.registry.queryDimensionRecords(‘visit’, dataId=data_ref.dataId.full)

```
visit:
  instrument: 'LSSTCam-imSim'
  id: 1192139
  physical_filter: 'r_sim_1.4'
  visit_system: 1
  name: '1192139'
  day_obs: 20261013
  exposure_time: 30.0
  target_name: 'UNKNOWN'
  observation_reason: 'imsim'
```

```
  science_program: '1192139'
  zenith_angle: 33.24096737781019
 region: ConvexPolygon([UnitVector3d(0.4540277748455429, 0.6654318675118485, -0.5925024973521197), UnitVector3d(0.4436108
 timespan: Timespan(begin=astropy.time.Time('2026-10-14 04:22:46.411200', scale='tai', format='iso'), end=astropy.time.Tim
```

Finally, repeat once more for “detector” info:

calexp_detector_metadata = butler.registry.queryDimensionRecords(‘detector’, dataId=data_ref.dataId.full)

```
detector:
  instrument: 'LSSTCam-imSim'
  id: 42
  full_name: 'R11_S20'
  name_in_raft: 'S20'
  raft: 'R11'
  purpose: 'SCIENCE'
```

We have thus demonstrated the ability to list and access metadata via the Butler.

Step 4 Step Execution Status: **Pass**

Description

Use ‘butler.get()‘ to directly retrieve a dataset with “metadata” in its DatasetType name (e.g., ‘characterizeImage_metadata’). Print the returned metadata values to the screen.

– –

Expected Result

Detailed information about the processing executed during the ‘characterizeImage‘ task is printed to the screen.

– –

Actual Result

To demonstrate a method of accessing a different type of metadata, execute the following:

char_image_metadata = butler.get(‘characterizeImage_metadata’, dataId = data_ref.dataId.full)

for v in char_image_metadata.values():

---

print(v)

```
runStartUtc = "2022-01-03T14:26:50.961867"
runStartCpuTime = 10567.533159602
runStartUserTime = 10185.162944000
runStartSystemTime = 382.37023900000
runStartMaxResidentSetSize = 3084460
runStartMinorPageFaults = 99345461
runStartMajorPageFaults = 8
runStartBlockInputs = 63352
runStartBlockOutputs = 78524216
runStartVoluntaryContextSwitches = 2236644
runStartInvoluntaryContextSwitches = 8650056
runEndUtc = "2022-01-03T14:26:50.967316"
runEndCpuTime = 10567.538608606
runEndUserTime = 10185.168389000
runEndSystemTime = 382.37023900000
runEndMaxResidentSetSize = 3084460
runEndMinorPageFaults = 99345461
runEndMajorPageFaults = 8
runEndBlockInputs = 63352
runEndBlockOutputs = 78524216
runEndVoluntaryContextSwitches = 2236644
runEndInvoluntaryContextSwitches = 8650056

...


doSmooth = 1
sigma = 1.8515346667980
smoothingKernelWidth = 13
nGrow = 4
detectFootprintsEndUtc = [ "2022-01-03T14:25:57.522184", "2022-01-03T14:26:16.980722" ]
detectFootprintsEndCpuTime = [ 10514.102591908, 10533.556957877 ]
detectFootprintsEndUserTime = [ 10132.464696000, 10151.463064000 ]
detectFootprintsEndSystemTime = [ 381.63792100000, 382.09391900000 ]
detectFootprintsEndMaxResidentSetSize = [ 3084460, 3084460 ]
detectFootprintsEndMinorPageFaults = [ 98951755, 99200762 ]
detectFootprintsEndMajorPageFaults = [ 8, 8 ]
detectFootprintsEndBlockInputs = [ 63352, 63352 ]
detectFootprintsEndBlockOutputs = [ 78524128, 78524152 ]
```

```
detectFootprintsEndVoluntaryContextSwitches = [ 2236644, 2236644 ]
detectFootprintsEndInvoluntaryContextSwitches = [ 8649887, 8649963 ]
runEndUtc = [ "2022-01-03T14:25:57.522993", "2022-01-03T14:26:16.981538" ]
runEndCpuTime = [ 10514.103401969, 10533.557777330 ]
runEndUserTime = [ 10132.465465000, 10151.463869000 ]
```

...

We show a portion of the results that are returned via 'butler.get()'. The full output is included in the attached notebook.

This has demonstrated two ways to list and access metadata via the Butler.

5.1.3.7 LVV-T32 - Verify implementation of Raw Image Assembly

Version **1**. Status **Approved**. Open *LVV-T32* test case in Jira.

Verify that the raw exposure data from all readout channels in a sensor can be assembled into a single image, and that all required/relevant metadata are associated with the image data.

Preconditions:

Execution status: **Pass**

Final comment:
The python script to execute this test is attached to the Test Report github repository in scripts/test_LVV-T32.py.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify a dataset with raw images that have been ingested into a butler repository.

Expected Result

Actual Result

For this test, we use weekly Science Pipelines version ‘w_2022_45‘ at the USDF, examining recently gathered AuxTel imaging data from the campaign detailed in SITCOM-505. The data were obtained in late October 2022, and processed at USDF with pipelines version ‘w_2022_44‘.

In particular, these data are at ‘/repo/embargo‘, in collection ‘u/huanlin/auxtel_oga_panda_test_2022102528_w44‘.

Step 2 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

from lsst.daf.butler import Butler

repo = ‘/repo/embargo’
collection = ‘u/huanlin/auxtel_oga_panda_test_2022102528_w44’

butler = Butler(repo, collections=collection)

Step 3 Step Execution Status: **Pass**

Description

Execute a butler query to identify and list raw images.

Expected Result

Actual Result

From the command line, issued the following query (note that the results are piped to an output file):
butler query-datasets /sdf/group/rubin/repo/embargo/ —collections 'u/huanlin/auxtel_oga_panda_test_2022102528_w44' raw > latiss_raws.txt

This returns 20550 raw AuxTel frames in a variety of filters. We select exposure 2022092900894, an r-band science exposure, for this test.

Step 4 Step Execution Status: **Pass**

Description

Retrieve one of the raw images returned by the query, and confirm that it is a well-formed image with reasonable pixel values.

Expected Result

A single raw image combining data from all readout channels for a given sensor.

Actual Result

The attached script retrieves the selected raw image via the following:

dataId = {'exposure':2022092900894, 'detector':0}
raw = butler.get('raw', dataId=dataId)

The script continues to verify that the image has expected properties, and prints the following to the screen:

What type of object is it?
<lsst.afw.image.exposure.ExposureF object at 0x7f784e357130>

Pixel data type:
<class 'numpy.float32'>

Visit info:
VisitInfo(exposureId=2022092900894, exposureTime=50, darkTime=50.2347, date=2022-09-30T08:18:54.069497388, UT1=nan, ERA=2.32734 rad, boresightRaDec=(48.4161301022, -30.2093423988), boresightAzAlt=(267.0042540913, +77.8991205080), boresightAirmass=1.02272, boresightRotAngle=6.28316 rad, rotType=1, observatory=-30.2446N, -70.7494E 2663, weather=Weather(nan, nan, nan), instrumentLabel='LATISS', id=2022092900894, focusZ=0.00603979, observationType='science', scienceProgram='SITCOM-479', observationReason='psf+photometry_validation', object='HD 20187', hasSimulatedContent=false)

Image shape and statistics:
Shape: (4096, 4608)
Mean, median, std deviation of pixel values: 14305.082 13714.0 3434.333

We have thus confirmed that the raw exposure data has been combined into a single "raw" image of the AuxTel detector, and has nonzero, reasonable pixel values.

Step 5 Step Execution Status: **Pass**

Description

Verify that a raw image was constructed with correct metadata.

Expected Result

Image header or ancillary table contains the required metadata about the observing context in which data were gathered.

Actual Result

In addition to the visitInfo shown in the previous step, the attached script also extracts the metadata attached to the image. The following code extracts the metadata and prints each entry to the screen:

```
md = raw.getMetadata()
md_dict = md.toDict()


for item in md_dict.items():
print(item)
```

The results are long; we thus truncate the results to demonstrate that basic information

```
Metadata:
('CCD_MANU', 'ITL')
('CCD_TYPE', '3800C')
('CCDGAIN', 1.0)
('CCDNOISE', 10.0)
('CCDSLOT', 'S00')
('RAFTBAY', 'R00')
('FIRMWARE', '11384004')
('PLATFORM', 'auxtel')
('CONTNUM', '189216ee')
('DAQVERS', 'R5-V0.6 2021-10-06T19:50:32Z (1800122)')
('DAQPART', 'lat')
('DAQFOLD', 'raw')
('SEQFILE', 'FP_ITL_2s_ir2_v25.seq')
('SEQNAME', 'FP_ITL_2s_ir2_v25.seq')
('SEQCKSUM', '2552520002')
('LSST_NUM', 'ITL-3800C-068')
('CCD_SERN', '20862')
('RAFTNAME', 'AuxTel-Raft')
('FPVERS', '1.1.4')
('IHVERS', '1.0.30')
('CCDTEMP', -90.7174)
('DATE', '2022-09-30T08:19:19.187')
('MJD', 59852.34674984962)
('IMGTYPE', 'OBJECT')
('DATE-OBS', '2022-09-30T08:18:28.952')
('MJD-OBS', 59852.346168425865)
('DATE-TRG', '2022-09-30T08:18:42.184')
('MJD-TRG', 59852.34632157395)
('OBSID', 'AT_O_20220929_000894')
('DATE-BEG', '2022-09-30T08:18:28.952')
('MJD-BEG', 59852.346168425865)
('DATE-END', '2022-09-30T08:19:19.187')
```

('MJD-END', 59852.34674984962)
('GROUPID', '2022-09-30T08:10:20.478')
('BUNIT', 'adu')
('TIMESYS', 'TAI')
('INSTRUME', 'LATISS')
('TELESCOP', 'LSST AuxTelescope')
('OBS-LONG', -70.749417)
('OBS-LAT', -30.244639)
('OBS-ELEV', 2663.0)
('OBSGEO-X', 1818938.94)
('OBSGEO-Y', -5208470.95)
('OBSGEO-Z', -3195172.08)
('FACILITY', 'Vera C. Rubin Observatory')
('RA', 48.41552041666668)
('DEC', -30.22323527777778)
('RASTART', 48.41613010222692)
('DECSTART', -30.209342398754412)
('RAEND', 48.41616329891139)
('DECEND', -30.20934820761029)
('ROTPA', 359.9987453403358)
('ROTCOORD', 'sky')
('HASTART', 0.9417104442537305)
('ELSTART', 77.89912050802377)
('AZSTART', -92.99574590867269)
('AMSTART', 1.0227214941767613)
('OBJECT', 'HD 20187')
('PROGRAM', 'SITCOM-479')
('REASON', 'PSF+Photometry_Validation')
('FILTBAND', 'r')
('FILTER', 'SDSSr')
('EXPTIME', 50.0)
('DARKTIME', 50.2347)
('SEEING', 0.9242383241653442)
('FILENAME', 'AT_O_20220929_000894_R00_S00.fits')

We have thus demonstrated that sufficient metadata about the observing context, telescope and instrument configuration, and observation details have been attached to the image.

5.1.3.8 LVV-T1756 - Verify calculation of photometric repeatability in uzy filters

Version **1**. Status **Approved**. Open *LVV-T1756* test case in Jira.

Verify that the DM system has provided the code to calculate the RMS photometric repeatability of bright non-saturated unresolved point sources in the u, z, and y filters, and assess whether it meets the requirement that it shall be less than **PA1uzy = 7.5 millimagnitudes**.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify a dataset containing at least one field in each of the u, z, and y filters with multiple overlapping visits.

Expected Result

A dataset that has been ingested into a Butler repository.

Actual Result

The HSC RC2 dataset described in DMTN-091 will be used for this test.

Step 2 Step Execution Status: **Pass**

Description

Execute ‘faro‘ on a repository containing processed data. Identify the path to the data, which we will call ‘DATA/-path’, then execute something similar to the following (with paths, datasets, and flags replaced or additionally specified as needed):

Example Code

```
pipetask –long-log run -j 2 -b DATA/path/butler.yaml –register-dataset-types -p $FARO_DIR/pipelines/metrics_pipeline.yaml -d “band in (’g’, ‘r’, ‘i’) AND tract=9813 AND skymap=’hsc_rings_v1’ AND instrument=’HSC’” –output u/username/-faro_metrics -i HSC/runs/RC2/w_2021_06 2>&1 | tee w06_2021_tract9813_faro.txt
```

Expected Result

The output collection (in this case, “u/username/faro_metrics”) containing metric measurements and any associated extras and metadata is available via the butler.

Actual Result

Rather than executing a specific processing pipeline, we demonstrate the calculation of metrics via the monthly reprocessing of RC2 data. In particular, the Characterization Metric Report for version 23 of the Science Pipelines (DMTR-351) summarizes the metrics that are typically monitored for pipeline characterization.

Step 3 Step Execution Status: **Pass**

Description

Confirm that the metric PA1uzy has been calculated, and that its values are reasonable.

Expected Result

A JSON file (and/or a report generated from that JSON file) demonstrating that PA1uzy has been calculated.

Actual Result

Section 4 of DMTR-351 summarizes photometric performance metrics, including photometric repeatability, PA1. The table (attached below) illustrates that the DM pipelines have provided software to calculate the photometric repeatability metric PA1. (Note that there are currently no u-band data, so repeatability cannot be calculated in u-band. However, the same software used for grizy bands will also work for u-band.)

Metric	Unit	SRD Requirement – Design	Release 22 Value (RC2)	Release 23 Value (RC2)	Comments
PA1: *u*	mmag	≤ 7.5	—	—	No data
PA1: *g*	mmag	≤ 5.0	7.6	7.1	
PA1: *r*	mmag	≤ 5.0	8.5	8.4	
PA1: *i*	mmag	≤ 5.0	9.2	8.7	
PA1: *z*	mmag	≤ 7.5	7.0	6.7	
PA1: *y*	mmag	≤ 7.5	8.0	7.9	
PF1: *u*	%	≤ 20	—	—	No data
PF1: *g*	%	≤ 20	12.0	10.7	
PF1: *r*	%	≤ 10	14.5	14.0	
PF1: *i*	%	≤ 10	16.0	14.5	
PF1: *z*	%	≤ 20	8.7	8.1	
PF1: *y*	%	≤ 10	12.0	11.5	

5.1.3.9 LVV-T199 - Verify implementation of Archive Center Co-Location with Existing Facility

Version **1**. Status **Approved**. Open *LVV-T199* test case in Jira.

Verify the Archive Center is located at an existing supported facility.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Analyze design

Expected Result

Actual Result

In RDO-18: PLAN for the OPERATIONS of the VERA C. RUBIN OBSERVATORY and Execution of its LEGACY SURVEY OF SPACE AND TIME, the executive summary states (emphasis added):

”This operational system will consist of facilities located at several distinct geographic sites—the Summit Facility on Cerro Pachón, Chile; the Base Facility in La Serena, Chile; *the US Data (processing and archiving) Facility at SLAC National Accelerator Laboratory (SLAC) in California;* the French Data Facility in Lyon, France; The UK Data Facility in Edinburgh, Scotland; and the headquarters facility in Tucson, Arizona–as well as the high-speed networks connecting these sites.”

...and in Section 2.2: An Operations Partnership:
“SLAC...will continue its managerial role as the lead DOE lab for operations and will expand its functional role to include camera maintenance and functional support, participation in assessing and assuring software, science, and the survey implementation, and operating the US Data Facility.”

The following text from Section 9.1.3: Operations Partners highlights the well-established, experienced nature of SLAC as a DOE lab:
“As the lead DOE lab, SLAC will represent and manage contributions from other DOE labs, including Fermilab, LLNL, and BNL... DOE personnel at SLAC and other laboratories also have extensive experience with data management and science assurance for large complex instruments and in other major sky survey projects (e.g., SDSS, Fermi, and DES) that will translate to Rubin Observatory operations.”

5.1.3.10 LVV-T190 - Verify implementation of Base Facility Co-Location with Existing Facility

Version **1**. Status **Approved**. Open *LVV-T190* test case in Jira.

Verify that the Base Facility is located at an existing known supported facility.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Analyze design

Expected Result

Actual Result
In RDO-18: PLAN for the OPERATIONS of the VERA C. RUBIN OBSERVATORY and Execution of its LEGACY SURVEY OF SPACE AND TIME, the executive summary states (emphasis added):

"This operational system will consist of facilities located at several distinct geographic sites—the Summit Facility on Cerro Pachón, Chile; *the Base Facility in La Serena, Chile;* the US Data (processing and archiving) Facility at SLAC National Accelerator Laboratory (SLAC) in California; the French Data Facility in Lyon, France; The UK Data Facility in Edinburgh, Scotland; and the headquarters facility in Tucson, Arizona–as well as the high-speed networks connecting these sites."

The following text from Section 4.6: Base Facility and Chilean Data Access Center makes it clear that the Base Facility is co-located with an existing facility supported by AURA's NOIRLab:

"The Base Facility is complete and is a new addition to AURA's NOIRLab La Serena office and workshop/lab complex called the recinto (in Spanish). The Base Facility consists of offices for the Observatory Operations department and a computing facility housed in a 5,000+ square-foot shared computing center. A dedicated, seismically resistant building with physical access control, hosts this computing center and supports the computer racks, power, ther-

mal conditioning, and network routing necessary for operations... A remote control room is part of the facility to allow remote scientific and technical support of operations on the summit. Summit monitoring equipment will be remotely operable from the Base Facility. The La Serena Base computing center will host the Chilean Data Access Center (DAC) and a backup data archive of the entire LSST data set... Through an agreement with the other AURA centers on the Recinto, the computing center also will be used to consolidate computing and network servers from Gemini, CTIO, SOAR, and AURA operations."

5.1.3.11 LVV-T77 - Verify implementation of Best Seeing Coadds

Version **1**. Status **Approved**. Open *LVV-T77* test case in Jira.

Verify that the DRP pipelines produce a suite of per-band coadds with input images filtered to optimize the size of the effective PSF on the coadd.

Preconditions:

Execution status: **Pass**

Final comment:
The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T77.py.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
The 'path' that you will use depends on where you are running the science pipelines. Options:

- local (newinstall.sh - based install):[path_to_installation]/loadLSST.bash

---

- development cluster (“lsst-dev”): /software/lsstsw/stack/loadLSST.bash
- LSP Notebook aspect (from a terminal): /opt/lsst/software/stack/loadLSST.bash

From the command line, execute the commands below in the example code:

Example Code

source ‘path‘
setup lsst_distrib

Expected Result

Science pipeline software is available for use. If additional packages are needed (for example, ‘obs’ packages such as ‘obs_subaru‘), then additional ‘setup‘ commands will be necessary.

To check versions in use, type:
eups list -s

Actual Result

Test executed on the RSP at the IDF. The test script confirms the pipelines version by printing the following to the screen:

lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup

Step 2 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

We use the DP0.2 data via the following:

```
config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)
```

Step 3 Step Execution Status: **Pass**

Description

Explicitly create a coadd for a specified seeing range in each filter.

Expected Result

Actual Result

For this test, we extract 'goodSeeingCoadd' data products that were produced during DP0.2 processing.

Step 4 Step Execution Status: **Pass**

Description

Verify that these coadds exist.

Expected Result

Actual Result

The test script loops over all ugrizy bands for a single tract/patch, extracts the 'goodSeeingCoadd', and verifies it is a well-formed image by printing various statistics to the screen. Finally, the script also extracts the 'deepCoadd_calexp' image and prints a comparison of the PSF radius to demonstrate that the best-seeing coadds have a smaller PSF radius than the deep coadds.

The output is as follows:

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'u'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.03438813 0.0024560345 5.8153605

Variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.020518279 0.019447705 0.10152267

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, 'NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}
Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'u'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'u'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 1.7244082609166302
Deep coadd PSF radius: 2.074471010117085

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'g'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.041773453 0.002788621 2.0373423

Variance plane shape and statistics:

Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.002616897 0.002469438 0.0104264915

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, 'NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}
Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'g'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'g'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 1.5164241757236603
Deep coadd PSF radius: 1.8854643031757852

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'r'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.08445272 0.0020215698 2.8918285

Variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.002568767 0.0024733383 0.011166171

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, '
NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}

Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'r'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'r'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 1.513700867128578
Deep coadd PSF radius: 1.8005040445327916

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'i'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.12454637 0.0037209666 4.126306

Variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: inf 0.009205212 nan

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, 'NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}
Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'i'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'i'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 1.5099309000064964
Deep coadd PSF radius: 1.728315912605641

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'z'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.15889181 0.012767363 5.4510126

Variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.082706526 0.080273375 0.040728822

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, 'NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}
Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'z'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'z'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 1.6839970224085898
Deep coadd PSF radius: 1.9398605388502517

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'y'}

Image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.2097747 0.022823093 7.169966

Variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.26777592 0.26288232 0.16389878

Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP': 2, 'NOT_DEBLENDED': 12, 'NO_DATA': 8, 'REJECTED': 13, 'SAT': 1, 'SENSOR_EDGE': 14, 'SUSPECT': 7, 'UNMASKEDNAN': 15}
Mask plane has dimensions: (4200, 4200)

WCS for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'y'} :
FITS standard SkyWcs:
Sky Origin: (55.6521739130, -31.9834710744)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

Photocalib for goodSeeingCoadd image with dataId {'tract': 4431, 'patch': 17, 'band': 'y'} :
spatially constant with mean: 57.544 error: 0

Good seeing coadd PSF radius: 2.2861573969027273
Deep coadd PSF radius: 2.508081993634421

5.1.3.12 LVV-T84 - Verify implementation of Bias Residual Image

Version **1**. Status **Approved**. Open *LVV-T84* test case in Jira.

Verify that DMS can construct a bias residual image that corrects for temporally-stable bias structures.
Verify that DMS can do this on demand.

Preconditions:

Execution status: **Pass**

Final comment:
Results are in the notebook “test_LVV-T84.ipynb” attached to the Test Report repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the location of an appropriate precursor dataset.

Expected Result

Actual Result
Working on the Rubin Science Platform (RSP) hosted at the USDF (SLAC), using Science Pipelines version w_2022_32.

The HSC/RC2 dataset we will use is in a shared repository at:
repo = ‘/sdf/group/rubin/repo/main’
collection = ‘HSC/runs/RC2/w_2022_28/DM-35609’

Step 2 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

butler = Butler(repo, collections=collection), where the repo and collection were specified in Step 1.

Step 3 Step Execution Status: **Pass**

Description

Import the standard libraries required for the rest of this test:

Example Code

```
import os
import lsst.afw.display as afwDisplay
from lsst.daf.persistence import Butler
from lsst.ip.isr import IsrTask
```

Expected Result

Actual Result

LIbraries imported.

Step 4 Step Execution Status: **Pass**

Description

Ingest the dataset from step 1 using the Butler (e.g., following example code below).

Example Code

```
butler = Butler($REPOSITORY_PATH)
raw = butler.get(тАЬrawтАЭ, visit=$VISIT_ID, detector=2)
bias = butler.get(тАЬbiasтАЭ, visit=$VISIT_ID, detector=2)
```

Expected Result

Actual Result

In our case, we specified a full dataId for a single i-band visit, then retrieved the corresponding raw and bias images:

dataId = {'instrument': 'HSC', 'detector': 42, 'visit': 30482, 'exposure':30482, 'band':'i'}
raw = butler.get('raw', dataId=dataId)
bias = butler.get('bias', dataId=dataId)

Step 5 Step Execution Status: **Pass**

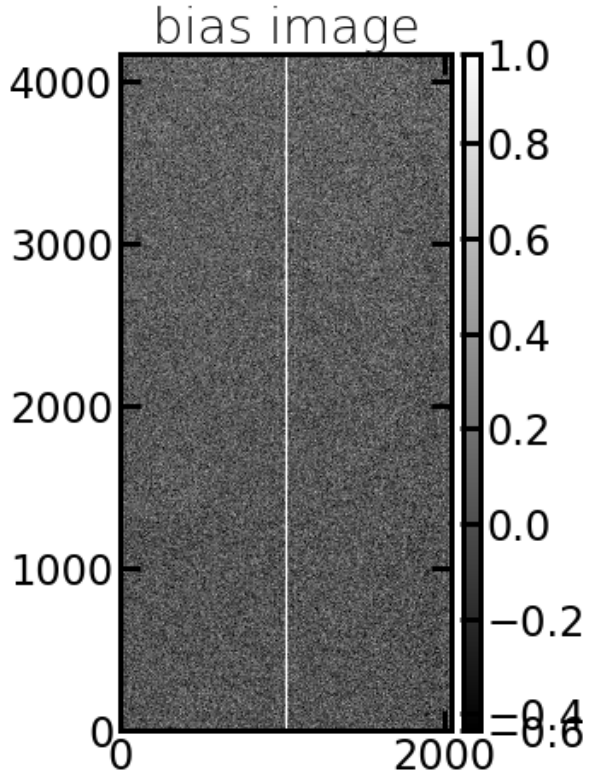
Description

Display the bias image and inspect that its pixels contain unique values.

Expected Result

A relatively flat image showing the bias level with roughly Poisson noise.

Actual Result



Step 6 Step Execution Status: **Pass**

Description

Configure and run an Instrument Signature Removal (ISR) task on the raw data. Most corrections are disabled for simplicity, but the bias frame is applied.

Example Code

```
isr_config = IsrTask.ConfigClass()
isr_config.doDark=False
isr_config.doFlat=False
isr_config.doFringe=False
isr_config.doDefect=False
isr_config.doLinearize=False
isr = IsrTask(config=isr_config)
result = isr.run(raw, bias=bias, detectorNum=raw.detector.getId(), camera=obs_lsst.LsstCamImSim.getCamera())
```

Expected Result

A trimmed, bias-corrected image in ‘result‘.

Actual Result

The “result” image was returned. To confirm the bias subtraction had the desired effect, we printed some statistics to the screen:

```
  image       median     stddev    (of image pixel values)
bias:        0.07311707 0.23739594
raw:         8130.0 2496.6898982587295
after bias:  6802.6895 895.51544
```

The noise is reduced significantly, suggesting the bias had its intended effect.

Step 7 Step Execution Status: **Pass**

Description

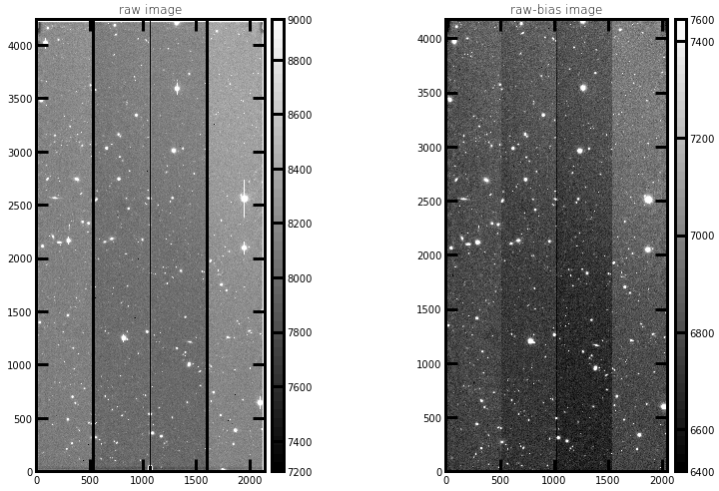
Display the ‘result‘ image and confirm that the bias correction has been performed.

Expected Result

A displayed image with bias removed (i.e., typical background counts reduced relative to the raw frame).

Actual Result

This image displays the raw frame on the left, and the same image after bias subtraction on the right.



We have thus demonstrated that the bias correction can be applied on demand.

5.1.3.13 LVV-T151 - Verify Implementation of Catalog Export Formats From the Notebook Aspect

Version **1**. Status **Approved**. Open *LVV-T151* test case in Jira.

Verify that catalog data is exportable from the notebook aspect in a variety of community-standard formats.

Preconditions:

Execution status: **Pass**

Final comment:

Results of the test execution can be found in the notebook “test_LVV-T151.ipynb” attached to

the github repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Authenticate to the notebook aspect of the Rubin Science Platform (NB-RSP). This is currently at either `https://data.lsst.cloud/nb` (for the interim data facility, or IDF) or `https://usdf-rsp.slac.stanford.edu/nb` (for the US data facility, or USDF).

Expected Result

Redirection to the spawner page of the NB-RSP allowing selection of the containerized science pipelines version and machine flavor.

Actual Result

Arrived at the spawner page.

Step 2 Step Execution Status: **Pass**

Description

Spawn a container by:
1) choosing an appropriate science pipelines version: e.g. the latest weekly.
2) choosing an appropriate machine flavor: e.g. medium
3) click “Spawn”

Expected Result

Redirection to the JupyterLab environment served from the chosen container containing the correct science pipelines version.

Actual Result

Spawned a “medium” container with Science Pipelines weekly version w_2022_32.

Step 3 Step Execution Status: **Pass**

Description

Open a new launcher by navigating in the top menu bar “File” -> “New Launcher”

Expected Result
A launcher window with several sections, potentially with several kernel versions for each.

Actual Result
Launcher appeared, with notebook, terminal, and other options.

Step 4 Step Execution Status: **Pass**

Description
Select the option under “Notebook” labeled “LSST” by clicking on the icon.

Expected Result
An empty notebook with a single empty cell. The kernel show up as “LSST” in the top right of the notebook.

Actual Result
Notebook appeared with w_2022_32 loaded, and an LSST kernel.

Step 5 Step Execution Status: **Pass**

Description
Execute a query in a notebook to select a small number of stars. In the example code below, we query the Data Preview 0.2 (DP0.2) catalog, then extract the results to an Astropy table.

Example Code

```
CELL 1:

from IPython.display import Markdown as md
from lsst.rsp import get_tap_service, retrieve_query

service = get_tap_service()
md(f'The service endpoint for TAP in this environment is:\n\n &#10145;     {service.baseurl}')

CELL 2:

results = service.search("SELECT coord_ra, coord_dec, g_cModelFlux, r_cModelFlux \
                          FROM dp02_dc2_catalogs.Object \
                          WHERE CONTAINS(POINT('ICRS', coord_ra, coord_dec), \
```

```
                    CIRCLE('ICRS', 60.0, -30.0, 0.05)) = 1")
```

Expected Result

Screen output from CELL 1:

The service endpoint for TAP in this environment is:
 `https://data.lsst.cloud/api/tap`

Example screen output from CELL 2 (may not contain the same 10 entries):

Table length=5533

coord_ra

coord_dec

g_cModelFlux

r_cModelFlux

deg

deg

nJy

nJy

float64

float64

float64

float64

59.9987401

-29.9728812

62.7060123

49.3496319

59.9995813

-29.9743232

166.0433743

394.8261645

59.9989853

-29.9750457

78.9557388

85.2691232

59.9993731

-29.9732406

111.0082072

165.6229656

60.0477786

-29.9736805

68.4818592

49.4783714

60.0400024

-29.9731507

52.0567337

114.2562171

60.0054666

-29.9728639

146.053072

134.1795803

60.00489

-29.9732239

1436.7150639

3606.8163133

60.0469583

-29.9735655

64.8838762

56.5677789

...

...

...

...

60.0053313

-30.0240394

125.6977786

379.8120713

59.9574061

-30.0163726

181.050889

200.8032979

60.0294415

-30.0241709

133.662163

230.8673464

59.9563419

-30.0239843

1551.2308712

4611.0406542

59.9879157

-30.0181116

76.3796313

46.5682713

60.0204061

-30.0228981

174.7738892

304.9991558

60.001638

-30.0183336

43.9593753

46.9695823

59.9861714

-30.0173405

164.6261404

288.8650875

59.9537443

-30.0160515

2228.7204658

5091.2041475

59.9683498

-30.0239539

835.415374

1101.0548649

—

—

Actual Result

The following was printed to the screen after CELL 1:

The service endpoint for TAP in this environment is:
 https://data.lsst.cloud/api/tap

After executing CELL 2, converted the results to an Astropy table using 'tab = results.to_table()'

Step 6 Step Execution Status: **Pass**

Description

Using the example code below, save the files to your storage space on the RSP Notebook Aspect.

Confirm that non-empty output files appear on disk.

Example Code

```
tab.write('test.csv', format='ascii.csv')
tab.write('test.vot', format='votable')
tab.write('test.fits', format='fits')
```

Expected Result

For the example given here, there should be the following files with the file size as listed:

- test.csv 5.7M
- test.vot 16M
- test.fits 4.5M

Actual Result

Upon executing, the three output files appear in the file browser on the left side of the screen. We checked the file output sizes in a Terminal window using 'ls -ltrh test.*'

Output:
-rw-r–r– 1 jeffcarlin 309K Aug 24 17:17 test.csv
-rw-r–r– 1 jeffcarlin 829K Aug 24 17:17 test.vot
-rw-r–r– 1 jeffcarlin 225K Aug 24 17:17 test.fits

Step 7 Step Execution Status: **Pass**

Description

Check that these files contain the same number of rows:

Example Code

```
from astropy.table import Table
dat_csv = Table.read('test.csv', format='ascii.csv')
dat_vot = Table.read('test.vot', format='votable')
dat_fits = Table.read('test.fits', format='fits')


import numpy as np
print(np.size(dat_csv), np.size(dat_vot), np.size(dat_fits))
```

Expected Result

Print statement produces output "5533 5533 5533".

Actual Result

As expected, the screen output was:

```
5533 5533 5533
```

Step 8 Step Execution Status: **Pass**

Description

Under the 'File' menu at the top of your Jupyter notebook session, select one of the following:

- Save All, Exit, and Log Out
- Exit and Log Out Without Saving

Expected Result

You will be returned to the RSP landing page: either `https://data.lsst.cloud/nb` (for the interim data facility, or IDF) or `https://usdf-rsp.slac.stanford.edu/nb` (for the US data facility, or USDF). It is now safe to close the browser window.

Actual Result
Successfully logged out and returned to the landing page.

5.1.3.14 LVV-T1232 - Verify Implementation of Catalog Export Formats From the Portal Aspect

Version **1**. Status **Approved**. Open *LVV-T1232* test case in Jira.

Verify that catalog data is exportable from the portal aspect in a variety of community-standard formats.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Navigate to the Portal Aspect endpoint. The stable version at the interim data facility (IDF) should be used for this test and is currently located at: `https://data.lsst.cloud/portal/app`.

Expected Result
A credential-entry screen should be displayed.

Actual Result
Clicked on the “login” button at upper right, and authenticated via github credentials.

Step 2 Step Execution Status: **Pass**

Description
Enter a valid set of credentials for an LSST user with RSP access on the instance under test.

Expected Result
The Portal Aspect UI should be displayed following authentication.

Actual Result
The Portal Aspect is displayed.

Step 3 Step Execution Status: **Pass**

Description
Select query type “ADQL”.

Expected Result

Actual Result
Selected “Edit ADQL” in box number 2. Box 1 default – “LSST RSP https://data.lsst.cloud/api/tap” is retained.

Step 4 Step Execution Status: **Pass**

Description
Execute the example query given in the example code below by entering the text in the ADQL Query box, then clicking “Search” at the lower left corner of the page.

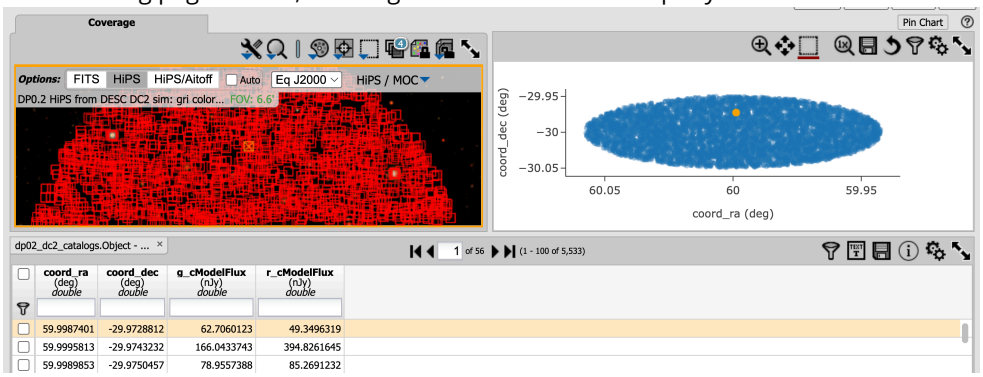
Example Code

```
SELECT coord_ra, coord_dec, g_cModelFlux, r_cModelFlux FROM dp02_dc2_catalogs.Object WHERE CONTAINS(POINT('ICRS', coord_ra,
```

Expected Result
A new page will load with the search results as a table, with some plots as well.

Actual Result

The following page loaded, showing 5533 results from the query:



Step 5 Step Execution Status: **Pass**

Description

Click the icon that looks like a floppy disk (it says “Save the content as an IPAC, CSV, or TSV table” when you mouse over it).

Expected Result

Actual Result

File save menu appeared.

Step 6 Step Execution Status: **Pass**

Description

- Select “CSV”, then specify a destination to save the file on your local computer.
- Select “VOTable”, then specify a destination to save the file on your local computer.
- Select “FITS”, then specify a destination to save the file on your local computer.

Expected Result

Actual Result

NOTE: the option to export as FITS-binary-table (i.e., .fits) is not currently supported by the Portal. We used “VOTable - FITS” for this execution, but if it is expected that FITS will be supported in the future, this test should be executed again

---

to verify FITS export.

After saving in all three formats to a local computer, executed the following:

'ls -ltrh test*'

Output:
-rw-r–r–@ 1 jcarlin staff 253K Aug 24 10:44 test_LVV-T1232_dp02_catalog.csv
-rw-r–r–@ 1 jcarlin staff 610K Aug 24 10:45 test_LVV-T1232_dp02_catalog.vot
-rw-r–r–@ 1 jcarlin staff 245K Aug 24 10:45 test_LVV-T1232_dp02_catalog.fits.vot

---

Step 7 Step Execution Status: **Pass**

Description
Open each of the files (either in TOPCAT, or using Astropy io tools). Confirm that the data tables are well-formed, and that each table contains the same columns and the same number of rows.

Expected Result

Actual Result
Opening each file in TOPCAT, we confirmed that all three of them have 5533 rows, and 4 columns.

Step 8 Step Execution Status: **Pass**

Description
Click the "logout" button at the upper right corner of the Portal screen.

Expected Result
Returned to the RSP home page at https://data.lsst.cloud/. When navigating to the portal endpoint, expect to execute the steps in LVV-T849.

Actual Result
Returned to the RSP home page.

5.1.3.15 LVV-T72 - Verify implementation of Coadd Image Method Constraints

Version **1**. Status **Approved**. Open *LVV-T72* test case in Jira.

Verify the implementation of how Coadd images are created.

Preconditions:

Execution status: **Pass**

Final comment:
The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T72.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify a dataset that has been processed to create coadd images.

Expected Result

Actual Result
We will use the Data Preview 0.2 dataset as processed at the Interim Data Facility (IDF).

Step 2 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working in a notebook entitled “test_LVV-T72.ipynb” on the Interim Data Facility at data.lsst.cloud:

```
from lsst.daf.butler import Butler

config = ‘dp02’
collection = ‘2.2i/runs/DP0.2’
butler = Butler(config, collections=collection)
```

Step 3 Step Execution Status: **Pass**

Description

Retrieve the coadds in the dataset and verify that they are non-empty.

Expected Result

Actual Result

First, we query the Butler to find all coadds overlapping the requested tract:

```
tract = 3828

data_refs_coadd = butler.registry.queryDatasets(
datasetType=“deepCoadd”,
where=f“tract={tract} and skymap=’DC2’”)
```

data_ids_coadd = []

for data_ref in data_refs_coadd:
data_ids_coadd.append(data_ref.dataId.full)

We examined a few images at random, and confirmed they are non-empty and contained all necessary associated data by printing the following output to the screen (see the notebook for more detail):

```
dataId:  {band: 'i', skymap: 'DC2', tract: 3828, patch: 1}

 Image shape and statistics:
Shape:  (4100, 4200)
Mean, median, std deviation of pixel values:  0.08778822 -0.00036619927 4.830672

 Variance plane shape and statistics:
Shape:  (4100, 4200)
Median of pixel values:  0.0033457286

 Mask plane has the following mask bits set:
{'BAD': 0, 'CLIPPED': 9, 'CR': 3, 'CROSSTALK': 10, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INEXACT_PSF': 11, 'INTRP
Mask plane has dimensions:  (4200, 4100)

Has PSF?  True

 WCS:
FITS standard SkyWcs:
Sky Origin: (56.6497461929, -36.4462809917)
Pixel Origin: (13999, 13999)
Pixel Scale: 0.2 arcsec/pixel

 Photocalib:
spatially constant with mean: 57.544 error: 0
```

Step 4 Step Execution Status: **Pass**

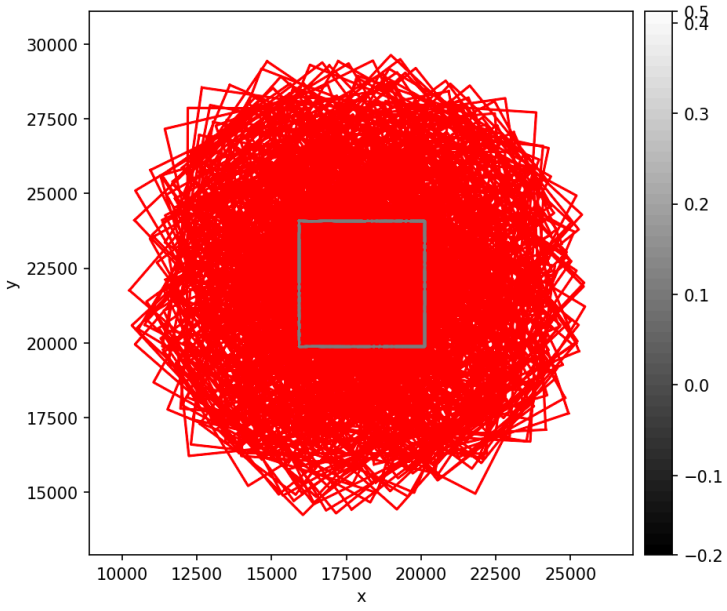
Description

Verify that coadds were created following specification

Expected Result

Actual Result

To verify that the coadds have been created using all of the overlapping visit images, we extract the catalog of overlapping images and plot their bounding boxes along with that of the coadd. Details can be seen in the attached notebook, which produced this image:



We have thus verified that the randomly-selected coadd was created following specification.

5.1.3.16 LVV-T90 - Verify implementation of Dark Current Correction Frame

Version **1**. Status **Approved**. Open *LVV-T90* test case in Jira.

Verify that the DMS can produce a dark correction frame calibration product.

Preconditions:

Execution status: **Pass**

Final comment:
Results are in the notebook “test_LVV-T90.ipynb” attached to the Test Report repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to a dataset containing dark frames (i.e., exposures taken with the shutter closed).

Expected Result

Actual Result
Working on the Rubin Science Platform (RSP) hosted at the USDF (SLAC), using Science Pipelines version w_2022_32.

The HSC/RC2 dataset we will use is in a shared repository at:
repo = ‘/sdf/group/rubin/repo/main’
collection = ‘HSC/runs/RC2/w_2022_28/DM-35609’

Step 2 Step Execution Status: **Pass**

Description
Execute the relevant steps from ‘cp_pipe‘ (the calibration pipeline) to produce dark correction frames.

Expected Result

Actual Result
The ISR correction steps have been executed during the bi-weekly RC2 reprocessing campaign.

Step 3 Step Execution Status: **Pass**

Description
Inspect the resulting dark correction frame to confirm that it appears as expected.

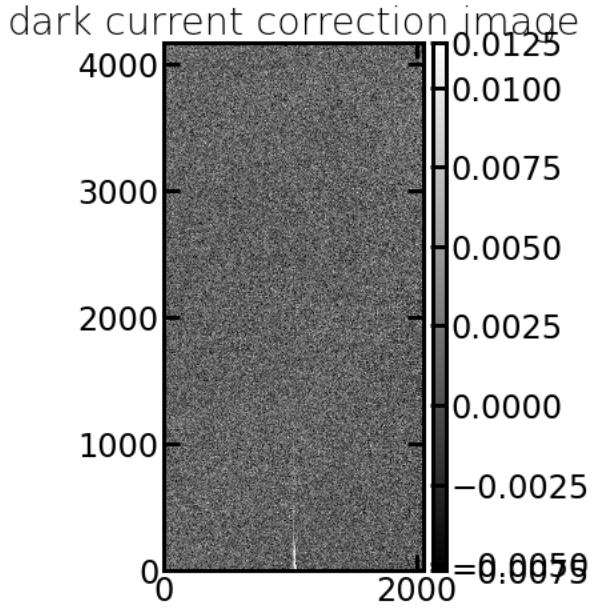
Expected Result

A well-formed dark correction frame is present and accessible via the Data Butler.

– –

Actual Result

The following image was retrieved from the Butler. It looks like a well-formed image, and shows some minor features and the bottom-middle region.



5.1.3.17 LVV-T137 - Verify implementation of Data Product Ingest

Version **1**. Status **Approved**. Open *LVV-T137* test case in Jira.

Verify that data products can be ingested.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify a suitable set of raw data to be run through “mini-DRP” processing.

Expected Result

Actual Result

This test will be executed using the rc2_subset data, which is a curated dataset selected to allow for full end-to-end processing of all DRP steps. The test will be executed on the USDF (SLAC) development machines, using Science Pipelines version w_2022_30.

Step 2 Step Execution Status: **Pass**

Description

Process data with the Data Release Production payload, starting from raw science images and generating science data products, placing them in the Data Backbone.

Expected Result

Actual Result

The shell script used to execute the steps of DRP processing is as follows:

```
# Set up Stack
# source /opt/lsst/software/stack/loadLSST.bash
source /cvmfs/sw.lsst.eu/linux-x86_64/lsst_distrib/w_2022_30/loadLSST.bash
setup lsst_distrib
eups list -s | grep lsst_distrib


#cd $HOME/DATA
cd /sdf/group/rubin/sandbox/$USER
#git clone https://github.com/lsst-dm/rc2_subset
cd repos
setup -j -r rc2_subset
```

```
echo $RC2_SUBSET_DIR

export NUMPROC=8

pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
-i HSC/RC2/defaults –register-dataset-types -o u/$USER/step1

echo "Running step2a on all visits"
pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
-i u/$USER/step1 –register-dataset-types -o u/$USER/step2

echo "Running step2b on tract 9813"
pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
-d "tract = 9813 and skymap = 'hsc_rings_v1'" –register-dataset-types -o u/$USER/step2

echo "Running step2c without multiprocessing"
pipetask –long-log run -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#nightlyStep2c'
–register-dataset-types -o u/$USER/step2

echo "Running step2d on all visits"
pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
–register-dataset-types -o u/$USER/step2

echo "Running step3 on all visits"
pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
-d "tract = 9813 and skymap = 'hsc_rings_v1' AND patch in (38, 39, 40, 41)" -i u/$USER/step2 –register-dataset-types
-o u/$USER/step3

echo "Running step4 on all visits"
pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p ${RC2_SUBSET_DIR}'/pipelines/DRP.yaml#ni
-d "tract = 9813 and skymap = 'hsc_rings_v1' AND patch in (38, 39, 40, 41)" -i u/$USER/step3 –register-dataset-types
-o u/$USER/step4
```

Step 3 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

See the attached script, "test_LVV-T137.py", which contains the following lines to initialize the Butler:

from lsst.daf.butler import Butler

repo = '/sdf/group/rubin/u/jcarlin/repos/rc2_subset/SMALL_HSC'
collection = 'u/jcarlin/step4'
butler = Butler(repo, collections=collection)

Step 4 Step Execution Status: **Pass**

Description

Confirm that the data products from the DRP processing have been ingested into the Data Backbone.

Expected Result

Processed images, catalogs, calibration information, and other related data products are present and accessible via the Butler.

Actual Result

The attached script, "test_LVV-T137.py" was executed from the command line via "python test_LVV-T137.py > test_LVV-T137.out". The screen output demonstrates that calibrated exposures have been created from ingested raws and calibration frames, and source detection and measurement has been performed (among other processing). The output of the script is (note: we have truncated the schema results for the "src" catalog for readability):

lsst_distrib g0b29ad24fb+434521fcbd current w_2022_35 setup
WCS for calexp image with dataId {'instrument': 'HSC', 'detector': 42, 'visit': 19680} :
FITS standard SkyWcs:
Sky Origin: (149.9670716242, +2.1456871368)
Pixel Origin: (989.716, 2019.07)
Pixel Scale: 0.168409 arcsec/pixel
Photocalib for calexp image with dataId {'instrument': 'HSC', 'detector': 42, 'visit': 19680} :
spatially constant with mean: 0.199449 error: 0.000171253
Image shape and statistics:
Shape: (4176, 2048)
Mean, median, std deviation of pixel values: 21.659693 3.1226554 277.56586
Variance plane shape and statistics:
Shape: (4176, 2048)
Mean, median, std deviation of pixel values: 918.03143 841.7631 476.36658
Mask plane has the following mask bits set:
{'BAD': 0, 'CR': 3, 'CROSSTALK': 9, 'DETECTED': 5, 'DETECTED_NEGATIVE': 6, 'EDGE': 4, 'INTRP': 2, 'NOT_DEBLENDED': 10, 'NO_DATA': 8, 'SAT': 1, 'SUSPECT': 7, 'UNMASKEDNAN': 11}
Mask plane has dimensions: (2048, 4176)
Source catalog has 5135 entries.
The first five entries in the source catalog:
id coord_ra ... calib_photometry_reserved
rad ...
——————- ——————— ... ———————————-
8452585832841217 2.615778319777679 ... False
8452585832841218 2.6157771306147075 ... False
8452585832841219 2.6157783826078296 ... False
8452585832841220 2.6157761384385596 ... False
8452585832841221 2.6157747119184185 ... False
The schema of the source catalog:
Schema(
(Field['L'](name="id", doc="unique ID"), Key<L>(offset=0, nElements=1)),
(Field['Angle'](name="coord_ra", doc="position in ra/dec"), Key<Angle>(offset=8, nElements=1)),
(Field['Angle'](name="coord_dec", doc="position in ra/dec"), Key<Angle>(offset=16, nElements=1)),
(Field['L'](name="parent", doc="unique ID of parent source"), Key<L>(offset=24, nElements=1)),
(Field['Flag'](name="calib_detected", doc="Source was detected as an icSource"), Key['Flag'](offset=32, bit=0)),
(Field['Flag'](name="calib_psf_candidate", doc="Flag set if the source was a candidate for PSF determination, as

determined by the star selector."), Key['Flag'](offset=32, bit=1)),
(Field['Flag'](name="calib_psf_used", doc="Flag set if the source was actually used for PSF determination, as determined by the"), Key['Flag'](offset=32, bit=2)),
(Field['Flag'](name="calib_psf_reserved", doc="set if source was reserved from PSF determination"), Key['Flag'](offset=32, bit=3)),
(Field['I'](name="deblend_nChild", doc="Number of children this object has (defaults to 0)"), Key<I>(offset=40, nElements=1)),
(Field['Flag'](name="deblend_deblendedAsPsf", doc="Deblender thought this source looked like a PSF"), Key['Flag'](offset=32, bit=4)),
(Field['D'](name="deblend_psfCenter_x", doc="If deblended-as-psf, the PSF centroid", units="pixel"), Key<D>(offset=48, nElements=1)),
(Field['D'](name="deblend_psfCenter_y", doc="If deblended-as-psf, the PSF centroid", units="pixel"), Key<D>(offset=56, nElements=1)),
(Field['D'](name="deblend_psf_instFlux", doc="If deblended-as-psf, the instrumental PSF flux", units="count"), Key<D>(offset=64, nElements=1)),

...

(Field['D'](name="ext_photometryKron_KronFlux_apCorrErr", doc="standard deviation of aperture correction applied to ext_photometryKron_KronFlux"), Key<D>(offset=992, nElements=1)),
(Field['Flag'](name="ext_photometryKron_KronFlux_flag_apCorr", doc="set if unable to aperture correct ext_photometryKron_KronF
Key['Flag'](offset=976, bit=1)),
(Field['D'](name="base_ClassificationExtendedness_value", doc="Set to 1 for extended sources, 0 for point sources."), Key<D>(offset=1000, nElements=1)),
(Field['Flag'](name="base_ClassificationExtendedness_flag", doc="Set to 1 for any fatal failure."), Key['Flag'](offset=976, bit=2)),
(Field['I'](name="base_FootprintArea_value", doc="Number of pixels in the source"s detection footprint.", units="pixel"), Key<I>(offset=1008, nElements=1)),
(Field['Flag'](name="calib_astrometry_used", doc="set if source was used in astrometric calibration"), Key['Flag'](offset=976, bit=3)),
(Field['Flag'](name="calib_photometry_used", doc="set if source was used in photometric calibration"), Key['Flag'](offset=976, bit=4)),
(Field['Flag'](name="calib_photometry_reserved", doc="set if source was reserved from photometric calibration"), Key['Flag'](offset=976, bit=5)),
'base_CircularApertureFlux_flag_badCentroid'->'base_SdssCentroid_flag'
'base_GaussianFlux_flag_badCentroid'->'base_SdssCentroid_flag'
'base_GaussianFlux_flag_badShape'->'ext_shapeHSM_HsmSourceMoments_flag'
'base_LocalBackground_flag_badCentroid'->'base_SdssCentroid_flag'
'base_NaiveCentroid_flag_badInitialCentroid'->'base_SdssCentroid_flag'
'base_PsfFlux_flag_badCentroid'->'base_SdssCentroid_flag'
'base_SdssShape_flag_badCentroid'->'base_SdssCentroid_flag'
'base_Variance_flag_badCentroid'->'base_SdssCentroid_flag'

'ext_photometryKron_KronFlux_flag_badInitialCentroid'->'base_SdssCentroid_flag'
'ext_shapeHSM_HsmPsfMomentsDebiased_flag_badCentroid'->'base_SdssCentroid_flag'
'ext_shapeHSM_HsmPsfMoments_flag_badCentroid'->'base_SdssCentroid_flag'
'ext_shapeHSM_HsmShapeRegauss_flag_badCentroid'->'base_SdssCentroid_flag'
'ext_shapeHSM_HsmSourceMomentsRound_flag_badCentroid'->'base_SdssCentroid_flag'
'ext_shapeHSM_HsmSourceMoments_flag_badCentroid'->'base_SdssCentroid_flag'
'slot_ApFlux'->'base_CircularApertureFlux_12_0'
'slot_CalibFlux'->'base_CircularApertureFlux_12_0'
'slot_Centroid'->'base_SdssCentroid'
'slot_GaussianFlux'->'base_GaussianFlux'
'slot_ModelFlux'->'base_GaussianFlux'
'slot_PsfFlux'->'base_PsfFlux'
'slot_PsfShape'->'ext_shapeHSM_HsmPsfMoments'
'slot_Shape'->'ext_shapeHSM_HsmSourceMoments'
)

Median of PSF magnitudes calculated using the associated PhotoCalib object:
24.88843561504042

5.1.3.18 LVV-T55 - Verify implementation of DIAForcedSource Catalog

Version **1**. Status **Approved**. Open *LVV-T55* test case in Jira.

Verify that the DMS produces a DIAForcedSource Catalog and that the catalog contains measured fluxes for DIAObjects.

Preconditions:

Execution status: **Pass**

Final comment:
The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T55.py.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Perform the steps of Alert Production (including, but not necessarily limited to, single frame processing, ISR, source detection/measurement, PSF estimation, photometric and astrometric calibration, difference imaging, DIASource detection/measurement, source association). During Operations, it is presumed that these are automated for a given dataset.

Expected Result

An output dataset including difference images and DIASource and DIAObject measurements.

Actual Result

For this test, we use the monthly reprocessing of the HSC RC2 dataset. All steps of end-to-end data processing are executed in the RC2 runs, including difference imaging.

Step 2 Step Execution Status: **Pass**

Description

Verify that the expected data products have been produced, and that catalogs contain reasonable values for measured quantities of interest.

Expected Result

Actual Result

Load and inspect the 'forcedSourceOnDiaObjectTable' to confirm that it was produced and all columns have been populated with measurements:

In [**11**]: forced_src = butler.get('forcedSourceOnDiaObjectTable_tract', tract=9615, skymap='hsc_rings_v1')

In [12]: forced_src
Out[12]:

forcedSourceOnDiaObjectId	diaObjectId	parentObjectId	coord_ra	coord_dec	ccdVisitId	...	pixelFlags_saturatedCenter	pixelFlags_crCenter	pixelFlags_suspectCenter	tract	patch
173516678758401	3425330564243128393	0	215.886840	0.254806	80800	...	False	False			

```
False  9615   15
173516678758402  3425330564243128394     0 215.889248  0.247669  80800 ...        False       False
False  9615   15
173516678758403  3425330564243128395     0 215.889644  0.241762  80800 ...        False       False
False  9615   15
173516678758404  3425330564243128399     0 215.897606  0.244158  80800 ...        False       False
False  9615   15

[39147253 rows x 34 columns]
```

The table looks well-formed.

Step 3 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working on the USDF (SLAC), a butler was instantiated by the following:

repo = '/sdf/group/rubin/repo/main'
collection = 'HSC/runs/RC2/w_2022_24/DM-35231'
butler = Butler(repo, collections=collection)

(i.e., pointing to processing of HSC RC2 data from weekly w_2022_24).

Step 4 Step Execution Status: **Pass**

Description

Confirm that the DIAForcedSource catalog contains measurements for each source, including DIAObject and visit IDs, the modeled flux and error, and measurement quality flags.

Expected Result

Actual Result

This test was executed via a script ("test_LVV-T55.py") that does the following:

- Execute a butler query to identify all visit/detector images overlapping a given tract/patch
- extract all 'forced_src' catalogs from the query results, and confirm that each has (a) forced photometry measurements for each DIAObject within its boundaries, and (b) the measurements consist of real numbers (i.e., not NaN or Inf).

These steps are then repeated for forced photometry on DiaObjects measured on 'calexps', then difference images. The code prints the following output:

```
$ python test_LVV-T55.py
lsst_distrib        g0b29ad24fb+a10408d0bf    w_2022_39 setup
100  of  358
200  of  358
300  of  358

Results for  334 visit/detectors that overlap the input tract.

DIA psfflux check:  True
DIA psfflux_err check:  True
DIA: All objects have forced sources?  True
100  of  358
200  of  358
300  of  358

Results for  334 visit/detectors that overlap the input tract.
```

DIA diffim psfflux check: True
DIA diffim psfflux_err check: True
DIA diffim: All objects have forced sources? True

This confirms that the forced source measurements have been performed for all DIAObjects, including on 'calexps' and 'diffims'.

5.1.3.19 LVV-T66 - Verify implementation of Forced-Source Catalog

Version **1**. Status **Approved**. Open *LVV-T66* test case in Jira.

Verify that all ForcedSources produced by the DRP pipelines contain fluxes measured on difference and direct single-epoch images, associated uncertainties, an Object ID, and a Visit ID.

Preconditions:

Execution status: **Pass**

Final comment:
The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T66.py.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

– –

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Working on the USDF (SLAC), a butler was instantiated by the following:

repo = '/sdf/group/rubin/repo/main'
collection = 'HSC/runs/RC2/w_2022_24/DM-35231'
butler = Butler(repo, collections=collection)

(i.e., pointing to processing of HSC RC2 data from weekly w_2022_24).

Step 2 Step Execution Status: **Pass**

Description

Retrieve the forced-source catalog from the Butler and verify it to be non-empty.

Expected Result

Actual Result

Result of printing the 'forced_src' catalog's first few rows to the screen:

id coord_ra coord_dec parent ... base_PsfFlux_flag_apCorr ext_photometryKron_KronFlux_apCorr ext_photometryKron_KronFlux_apCorrErr ext_photometryKron_KronFlux_flag_apCorr
rad rad ...
---- ---- ---- --- ... ---- ---- ---- ----
5029488307994625 2.63104145229885 0.038054902578442455 0 ... False 1.0270320546384992

0.0 False
5029488307994626 2.630928461400815 0.037891765102001845 0 ... False 1.026189759742002
0.0 False
5029488307994627 2.632233427600064 0.03791343935244524 0 ... False 1.02175858713207
0.0 False
5029488307994628 2.6304566006364767 0.03788995502872067 0 ... False 1.0266876839647747
0.0 False
5029488307994629 2.630538198528984 0.03788277899472417 0 ... False 1.0265930273455812
0.0 False
5029488307994630 2.6318566450014282 0.03790571746997802 0 ... False 1.0235191928776037
0.0 False
5029488307994631 2.6314240224499974 0.03795431495760043 0 ... False 1.0254220348170437
0.0 False
5029488307994632 2.631721634971245 0.037888599928408165 0 ... False 1.0239559189552083
0.0 False
5

The catalog exists, and all columns are populated with measurements (we will further verify this in subsequent steps).

Step 3 Step Execution Status: **Pass**

Description

Verify that there exist entries in the forced-photometry table for all coadd objects for the PVIs on which the object should appear.

Expected Result

Actual Result

This test was executed via a script ("test_LVV-T66.py") that does the following:

- Execute a butler query to identify all visit/detector images overlapping a given tract/patch
- Load objectIds from the 'objectTable_tract' for the input tract
- extract all 'forced_src' catalogs from the query results, and confirm that each has (a) forced photometry measurements for each object within its boundaries, and (b) the measurements consist of real numbers (i.e., not NaN or Inf).

These steps are then repeated for forced photometry on DiaObjects measured on 'calexps', then difference images. The code prints the following output:

```
$ python test_LVV-T66.py
lsst_distrib        g0b29ad24fb+a10408d0bf    current w_2022_39 setup

Results for  334 visit/detectors that overlap the input tract.

psfflux check:  True
psfflux_err check:  True
apflux check:  True
apflux_err check:  True
All objects have forced sources?  True

Results for  334 visit/detectors that overlap the input tract.

DIA psfflux check:  True
DIA psfflux_err check:  True
DIA: All objects have forced sources?  True

Results for  334 visit/detectors that overlap the input tract.

DIA diffim psfflux check:  True
DIA diffim psfflux_err check:  True
DIA diffim: All objects have forced sources? True
```

This confirms that the forced source measurements have been performed for all objects, including on ‘calexps‘ and ‘diffims‘.

Step 4 Step Execution Status: **Pass**

Description

Verify that there exist entries in a forced-photometry table for each image for all DIAObjects.

Expected Result

Actual Result
This was included in the attached script (see the previous test step).

5.1.3.20 LVV-T91 - Verify implementation of Fringe Correction Frame

Version **1**. Status **Approved**. Open *LVV-T91* test case in Jira.

Verify that the DMS can produce an fringe-correction frame calibration product.
Verify that the DMS can determine the effectiveness of the fringe-correction frame and determine how often it should be updated.

Preconditions:

Execution status: **Pass**

Final comment:
Results are in the notebook “test_LVV-T91.ipynb” attached to the Test Report repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Execute Test Case LVV-T88, which runs the calibration products pipeline.

Expected Result

Actual Result
Working on the Rubin Science Platform (RSP) hosted at the USDF (SLAC), using Science Pipelines version w_2022_32.

The HSC/RC2 dataset we will use is in a shared repository at:
repo = '/sdf/group/rubin/repo/main'
collection = 'HSC/runs/RC2/w_2022_28/DM-35609'

The ISR correction steps have been executed during the bi-weekly RC2 reprocessing campaign.

Step 2 Step Execution Status: **Pass**

Description

Examine the fringe-correction frames created by the pipeline to ensure that they are well-formed.

Expected Result

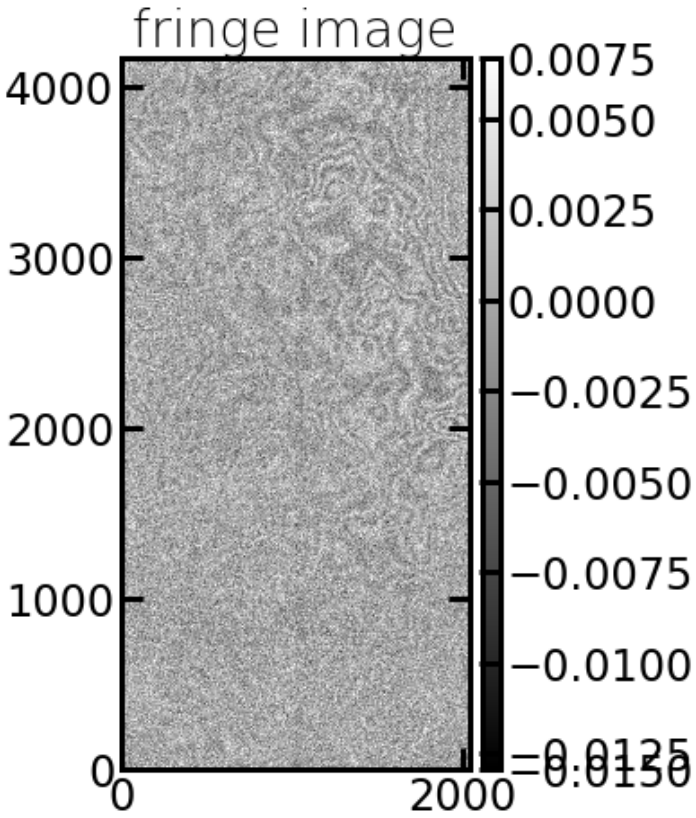
Fringe frame is an lsst.afw.image.Exposure with reasonable pixel values.

Actual Result

We selected and examined the image with the following dataId:

dataId = {'instrument': 'HSC', 'detector': 42, 'visit': 1880, 'exposure': 1880}

The image has reasonable statistics, and displays the characteristic fringing patterns:



Step 3 Step Execution Status: **Pass**

Description

Apply the fringe correction to a science image and confirm that it has the desired effect.

Expected Result

Images before and after correction have different pixel values.

Actual Result

In the attached notebook, we extracted image statistics before and after the fringe correction, and see the following:

```
  image                    median     stddev    (of image pixel values)
raw:                     17078.0 3836.131470952384
```

```
after fringe correction:  17108.0 765.4047
```

We have thus demonstrated that the fringe correction has altered the image as expected.

5.1.3.21 LVV-T126 - Verify implementation of Image Differencing

Version **1**. Status **Approved**. Open *LVV-T126* test case in Jira.

Verify that the DMS can perform image differencing from single exposures and coadds.

Preconditions:

Execution status: **Pass**

Final comment:
Results are in the notebook “test_LVV-T126.ipynb” attached to the Test Report repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify a repository containing data that have been processed through the difference imaging pipeline. (e.g., the HiTS 2015 data that are processed monthly for testing)

Expected Result
A dataset containing calexps, difference images, and source catalogs (of diaSrcs).

Actual Result
We execute this test on the RSP at the IDF, using DP0.2 data and Science Pipelines version w_2022_32.

Step 2 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

To instantiate the butler containing DP0.2 data:

config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)

Step 3 Step Execution Status: **Pass**

Description

Extract a 'calexp', a 'deepDiff_differenceExp', and the 'deepDiff_diaSrc' catalog of measurements.

Expected Result

Well-formed images and catalogs containing the calexp from the visit image and the difference image, and measurements of sources from the difference image.

Actual Result

Select the dataId of a single visit image:
dataId = {'instrument': 'LSSTCam-imSim', 'detector': 78, 'visit': 60891, 'exposure':60891, 'band':'i'}

Extract the calexp image, difference image, and diaSrc catalog:
calexp = butler.get('calexp', dataId=dataId)
diffim = butler.get('goodSeeingDiff_differenceExp', dataId=dataId)
diasrc = butler.get('goodSeeingDiff_diaSrc', dataId=dataId)

Step 4 Step Execution Status: **Pass**

Description

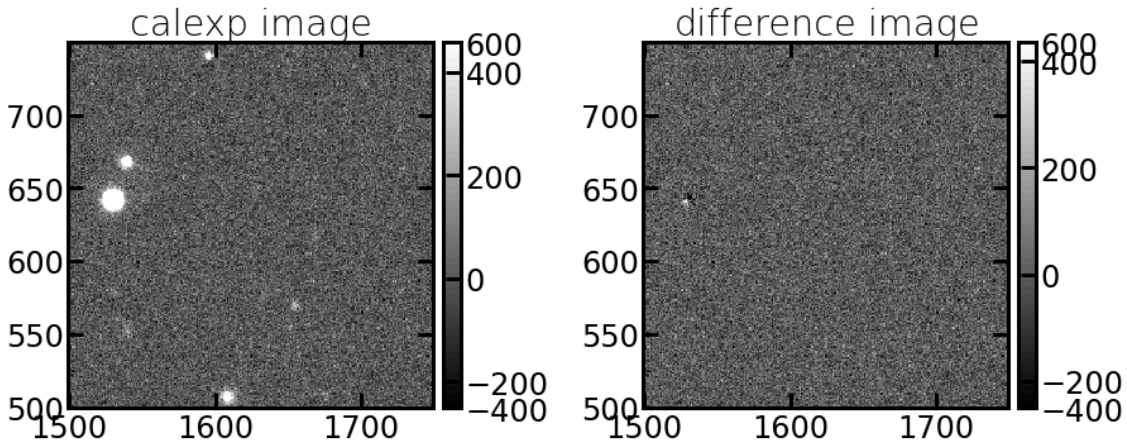
Confirm (by visual inspection) that the difference image is mostly blank sky (i.e., has had a template of the same field subtracted), and that the source catalog contains sources with photometric and astrometric measurements.

Expected Result

A mostly blank image (with perhaps some artifacts due to imperfect subtraction) and a catalog of sources detected/measured from that image.

Actual Result

This following figure shows the calexp image on the left and the difference image for the same visit on the right. The difference image contains mostly blank sky, with some minor artifacts due to imperfect subtraction of bright stars.



The attached notebook demonstrates that the diaSrc catalog contains the expected measurements.

5.1.3.22 LVV-T1946 - Verify implementation of measurements in catalogs from coadds

Version **1**. Status **Approved**. Open *LVV-T1946* test case in Jira.

Verify that source measurements in catalogs containing measurements from coadd images are in flux units.

Preconditions:

Execution status: **Pass**

Final comment:
Executed at the IDF using Science Pipelines version w_2022_32.

This test case can be executed by running the script test_LVV-T1946.py, which is available in the test report github repository's "scripts/" directory.

Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Tests were performed by executing test_LVV-T1946.py, which contains the following:

```
# For DP0.2 data on the IDF:
config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)
```

Step 2 Step Execution Status: **Pass**

Description

Identify and read an appropriate processed precursor dataset containing coadds with the Butler.

Expected Result

Actual Result

By default, the test script contains the following tract/patch/band combination. This can be changed if desired.

```
# Select an arbitrary source catalog from a deepCoadd:
band = 'i'
tract = 3828
patch = 13

# Run the test
LVVT1946(tract, patch, band)
```

Step 3 Step Execution Status: **Pass**

Description

Verify that the coadd catalog provides measurements in flux units.

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

Execute the script, which checks all flux fields for the input coadd image:
'python test_LVV-T1946.py'

Screen outputs:
lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup
Input dataId: {'tract': 3828, 'band': 'i', 'patch': 13}
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count
base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count
base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_LocalBackground_instFlux ..... count
base_LocalBackground_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_0_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_0_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_0_7_instFlux ..... count
ext_gaap_GaapFlux_1_15x_0_7_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_1_0_instFlux ..... count

ext_gaap_GaapFlux_1_15x_1_0_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_1_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_1_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_2_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_2_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_3_0_instFlux ..... count
ext_gaap_GaapFlux_1_15x_3_0_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_PsfFlux_instFlux ..... count
ext_gaap_GaapFlux_1_15x_PsfFlux_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_Optimal_instFlux ..... count
ext_gaap_GaapFlux_1_15x_Optimal_instFluxErr ..... count
ext_photometryKron_KronFlux_instFlux ..... count
ext_photometryKron_KronFlux_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_0_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_0_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_0_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_0_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_1_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_1_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_1_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_1_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_2_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_2_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_2_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_2_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_3_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_3_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_3_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_kron_instFlux ..... count

ext_convolved_ConvolvedFlux_3_kron_instFluxErr ..... count
modelfit_CModel_initial_instFlux ..... count
modelfit_CModel_initial_instFluxErr ..... count
modelfit_CModel_initial_instFlux_inner ..... count
modelfit_CModel_exp_instFlux ..... count
modelfit_CModel_exp_instFluxErr ..... count
modelfit_CModel_exp_instFlux_inner ..... count
modelfit_CModel_dev_instFlux ..... count
modelfit_CModel_dev_instFluxErr ..... count
modelfit_CModel_dev_instFlux_inner ..... count
modelfit_CModel_instFlux ..... count
modelfit_CModel_instFluxErr ..... count
modelfit_CModel_instFlux_inner ..... count
undeblended_base_CircularApertureFlux_3_0_instFlux ..... count
undeblended_base_CircularApertureFlux_3_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_4_5_instFlux ..... count
undeblended_base_CircularApertureFlux_4_5_instFluxErr ..... count
undeblended_base_CircularApertureFlux_6_0_instFlux ..... count
undeblended_base_CircularApertureFlux_6_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_9_0_instFlux ..... count
undeblended_base_CircularApertureFlux_9_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_12_0_instFlux ..... count
undeblended_base_CircularApertureFlux_12_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_17_0_instFlux ..... count
undeblended_base_CircularApertureFlux_17_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_25_0_instFlux ..... count
undeblended_base_CircularApertureFlux_25_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_35_0_instFlux ..... count
undeblended_base_CircularApertureFlux_35_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_50_0_instFlux ..... count
undeblended_base_CircularApertureFlux_50_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_70_0_instFlux ..... count
undeblended_base_CircularApertureFlux_70_0_instFluxErr ..... count
undeblended_base_PsfFlux_instFlux ..... count
undeblended_base_PsfFlux_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_5_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_7_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_7_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_0_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_0_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_5_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_2_5_instFlux ..... count

undeblended_ext_gaap_GaapFlux_1_15x_2_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_3_0_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_3_0_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_PsfFlux_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_PsfFlux_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_Optimal_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_Optimal_instFluxErr ..... count
undeblended_ext_photometryKron_KronFlux_instFlux ..... count
undeblended_ext_photometryKron_KronFlux_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_kron_instFluxErr ..... count

All forced_src instFlux entries have units of counts: True

All flux fields have units of “counts”, so this test has passed.

5.1.3.23 LVV-T1947 - Verify implementation of measurements in catalogs from difference images

Version **1**. Status **Approved**. Open *LVV-T1947* test case in Jira.

Verify that source measurements in catalogs containing measurements from difference images are in flux units.

Preconditions:

Execution status: **Pass**

Final comment:
Executed at the IDF using Science Pipelines version w_2022_32.

This test case can be executed by running the script test_LVV-T1947.py, which is available in the test report github repository’s “scripts/” directory.

Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Tests were performed by executing test_LVV-T1947.py, which contains the following:

```
# For DP0.2 data on the IDF:
config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)
```

Step 2 Step Execution Status: **Pass**

Description

Identify and read an appropriate processed precursor dataset containing difference images with the Butler.

Expected Result

Actual Result

By default, the test script contains the following tract/patch/band combination. This can be changed if desired.

```
# Select an arbitrary source catalog from a difference image:
instrument = 'LSSTCam-imSim'
detector = 5
visit = 924041
```

```
# Run the test
```

LVVT1947(instrument, visit, detector)

Step 3 Step Execution Status: **Pass**

Description

Verify that the difference image source catalog provides measurements in flux units.

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

Execute the script, which checks all flux fields for the input goodSeeingDiff image:
'python test_LVV-T1947.py'

Screen outputs:

```
lsst_distrib        g0b29ad24fb+cafeaf151e    current w_2022_32 setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count
base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count
base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
```

base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_PeakLikelihoodFlux_instFlux ..... count
base_PeakLikelihoodFlux_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ip_diffim_NaiveDipoleFlux_pos_instFlux ..... count
ip_diffim_NaiveDipoleFlux_pos_instFluxErr ..... count
ip_diffim_NaiveDipoleFlux_neg_instFlux ..... count
ip_diffim_NaiveDipoleFlux_neg_instFluxErr ..... count
ip_diffim_PsfDipoleFlux_pos_instFlux ..... count
ip_diffim_PsfDipoleFlux_pos_instFluxErr ..... count
ip_diffim_PsfDipoleFlux_neg_instFlux ..... count
ip_diffim_PsfDipoleFlux_neg_instFluxErr ..... count
ip_diffim_DipoleFit_pos_instFlux ..... count
ip_diffim_DipoleFit_pos_instFluxErr ..... count
ip_diffim_DipoleFit_neg_instFlux ..... count
ip_diffim_DipoleFit_neg_instFluxErr ..... count
ip_diffim_DipoleFit_instFlux ..... count
ip_diffim_forced_PsfFlux_instFlux ..... count
ip_diffim_forced_PsfFlux_instFluxErr ..... count

All diaSrc table instFlux entries have units of counts: True

All flux fields have units of “counts”, so this test has passed.

5.1.3.24 LVV-T28 - Verify implementation of measurements in catalogs from PVIs

Version **1**. Status **Approved**. Open *LVV-T28* test case in Jira.

Verify that source measurements in catalogs containing measurements from processed visit images are in flux units.

Preconditions:

Execution status: **Pass**

Final comment:
Executed at the IDF using Science Pipelines version w_2022_32.

This test case can be executed by running the script test_LVV-T28.py, which is available in the test report github repository's "scripts/" directory.

Note that in this test we only checked source catalogs that are persisted in butler repositories. Catalogs in other forms (e.g., parquet tables, or tables in the Qserv database) will be confirmed separately.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result
Butler repo available for reading.

Actual Result
Tests were performed by executing test_LVV-T28.py, which contains the following:

# For DP0.2 data on the IDF:
config = 'dp02'
collection = '2.2i/runs/DP0.2'

---

butler = Butler(config, collections=collection)

Step 2 Step Execution Status: **Pass**

Description

Identify and read an appropriate processed precursor dataset containing coadds with the Butler.

Expected Result

Actual Result

By default, the test script contains the following tract/patch/band combination. This can be changed if desired.

# Select an arbitrary source catalog from a deepCoadd:
instrument = 'LSSTCam-imSim'
detector = 5
visit = 924041

# Run the test
LVVT28(instrument, visit, detector)

Step 3 Step Execution Status: **Pass**

Description

Verify that the single-visit catalog provides measurements in flux units.

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

Execute the script, which checks all flux fields for the input PVI:
'python test_LVV-T28.py'

lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}
deblend_psf_instFlux ..... count
base_Blendedness_raw_child_instFlux ..... count

---

base_Blendedness_raw_parent_instFlux ..... count
base_Blendedness_abs_child_instFlux ..... count
base_Blendedness_abs_parent_instFlux ..... count
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count
base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count
base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_LocalBackground_instFlux ..... count
base_LocalBackground_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ext_photometryKron_KronFlux_instFlux ..... count
ext_photometryKron_KronFlux_instFluxErr ..... count

All src table instFlux entries have units of counts: True

All flux fields have units of “counts”, so this test has passed.

5.1.3.25 LVV-T78 - Verify implementation of Persisting Data Products

Version **1**. Status **Approved**. Open *LVV-T78* test case in Jira.

Verify that per-band deep coadds and best-seeing coadds are present, kept, and available.

Preconditions:
Precursor data from HSC PDR or simulated DC2 data (as reprocessed with LSST Science Pipelines to produce Data Preview 0.2).

Execution status: **Pass**

Final comment:
The python test script to execute this code is attached to the Test Report github repository in scripts/test_LVV-T78.py.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result
Butler repo available for reading.

Actual Result
Test executed on the RSP at the IDF. The test script confirms the pipelines version by printing the following to the screen:

lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup

Step 2 Step Execution Status: **Pass**

Description

Identify some single-band deep coadds and retrieve them from the butler

Expected Result

Actual Result

We use the DP0.2 data via the following:

config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)

Step 3 Step Execution Status: **Pass**

Description

Examine the deep coadds and confirm that they are well-formed images

Expected Result

Actual Result

All subsequent steps were executed as part of the python script test_LVV-T78.py. The results are posted below.

Step 4 Step Execution Status: **Pass**

Description

Identify some single-band best-seeing coadds and retrieve them from the butler

Expected Result

Actual Result

Step 5 Step Execution Status: **Pass**

Description

Examine the best-seeing coadds and confirm that they are well-formed images

Expected Result

Actual Result

The output of the python script is as follows. This script reads both a best-seeing coadd and a deep coadd corresponding to the same dataId, prints statistics of each image to the screen, examines the variance plane and mask to confirm they are well-formed, and confirms that each image has an associated PSF and WCS.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'u'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.03438813 0.0024560345 5.8153605

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.04503397 0.0047773793 7.005774

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.020518279 0.019447705 0.10152267

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.0056145624 0.0053308182 0.14121237
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'g'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.041773453 0.002788621 2.0373423

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.05321344 0.0066212164 2.2148554

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.002616897 0.002469438 0.0104264915

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.00074665045 0.00070681534 0.0032021273
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'r'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.08445272 0.0020215698 2.8918285

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.10712123 0.009368613 3.528262

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.002568767 0.0024733383 0.011166171

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.00093781634 0.0008518264 0.012951709
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'i'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.12454637 0.0037209666 4.126306

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.15418924 0.0157091 4.6056767

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: inf 0.009205212 nan

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: inf 0.0031147646 nan
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'z'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.15889181 0.012767363 5.4510126

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.18890294 0.026478318 5.2928886

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.082706526 0.080273375 0.040728822

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.026652763 0.026110252 0.019691393
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

Input dataId: {'tract': 4431, 'patch': 17, 'band': 'y'}

Best-seeing coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.2097747 0.022823093 7.169966

Deep coadd image shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.26832125 0.039412603 10.62081

Best-seeing coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.26777592 0.26288232 0.16389878

Deep coadd variance plane shape and statistics:
Shape: (4200, 4200)
Mean, median, std deviation of pixel values: 0.08693949 0.08555089 0.26010248
Best-seeing coadd mask plane has dimensions: (4200, 4200)
Deep coadd mask plane has dimensions: (4200, 4200)
Both images have a WCS.
Both images have a PSF.

We have thus verified that deep and best-seeing coadds are present, retained, and available, and are well-formed.

5.1.3.26 LVV-T42 - Verify implementation of Processed Visit Image Content

Version **1**. Status **Approved**. Open *LVV-T42* test case in Jira.

Verify that Processed Visit Images produced by the DRP and AP pipelines include the observed data, a mask array, a variance array, a PSF model, and a WCS model.

Preconditions:

Execution status: **Pass**

Final comment:
Results are in the notebook “test_LVV-T42.ipynb” attached to the Test Report repository.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result
Butler repo available for reading.

Actual Result

On the IDF, working with Science Pipelines version w_2022_32, executed the following to initialize the Butler with the DP0.2 dataset:

```
# For DP0.2 data on the IDF:
config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)
```

Step 2 Step Execution Status: **Pass**

Description

Ingest the data from an appropriate processed dataset.

Expected Result

Actual Result

Successfully initialized the butler in the previous step.

Step 3 Step Execution Status: **Pass**

Description

Select a single visit from the dataset, and extract its WCS object, calexp image, psf model, and source list.

Expected Result

Actual Result

Select a single visit with this dataId:

dataId = {'instrument': 'LSSTCam-imSim', 'detector': 78, 'visit': 60891, 'exposure':60891, 'band':'i'}

Extract the calexp image, source list, psf model, and WCS object,

```
calexp = butler.get('calexp', dataId=dataId)
src = butler.get('src', dataId=dataId)
```

```
psf = calexp.getPsf()
wcs = calexp.getWcs()
```

Step 4 Step Execution Status: **Pass**

Description

Inspect the calexp image to ensure that

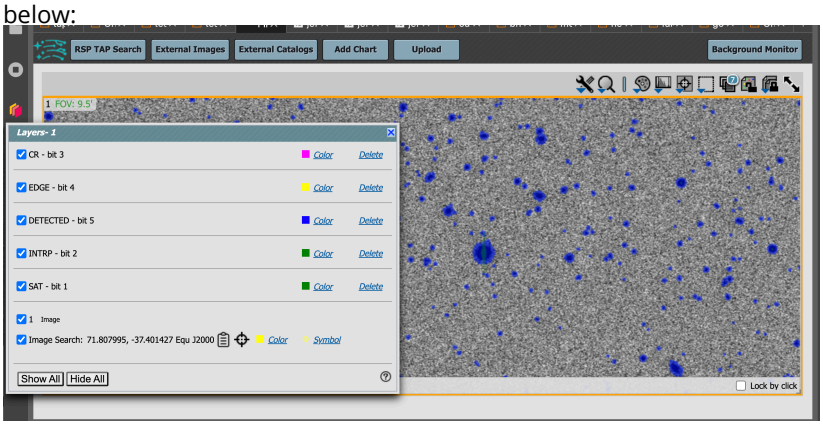
1. A well-formed image is present,
2. The variance plane is present and well-behaved,
3. Mask planes are present and contain information about defects.

Expected Result

An astronomical image with mask and variance planes. This can be readily visualized using Firefly, which displays mask planes by default.

Actual Result

We inspected the calexp and its associated image planes in Firefly, confirming that the variance plane and image plane differ, and that they are well-formed images. An example showing the mask plane overlaid on the calexp is below:



The above image also shows the definitions of the different-colored mask bits that are set.

Step 5 Step Execution Status: **Pass**

Description

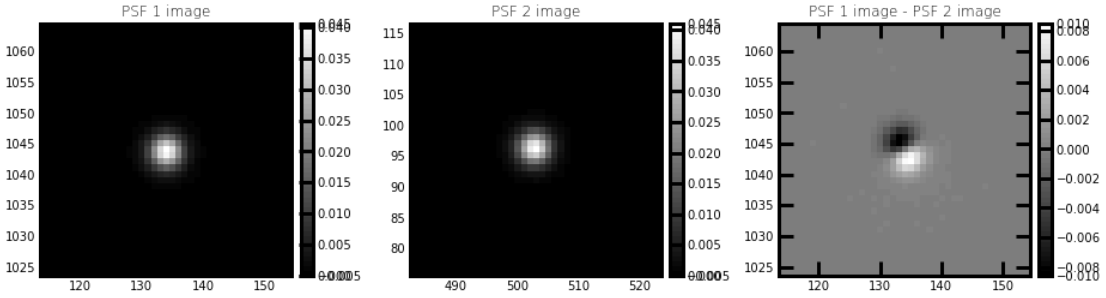
Plot images of the PSF model at various points, and verify that the PSF differs with position.

Expected Result

A "star-like" image of the PSF evaluated at various positions. The PSF should vary slightly with position (this could be readily visualized by taking a difference of PSFs at two positions).

Actual Result

The following image shows the PSF extracted at two different positions in the first two panels, and their difference in the third. The PSF is clearly varying with position:



Step 6 Step Execution Status: **Pass**

Description

Starting from the XY pixel coordinates of the sources, apply the WCS to obtain RA, Dec coordinates. Plot these positions and confirm that they match the expected values from the WCS object.

Expected Result

RA, Dec coordinates that are returned should be near the central position of the visit coordinate as given in either the calexp metadata or the WCS.

Actual Result

We selected a single source from the catalog at positions:

```
(X, Y): ( 1730.342687466467 ,  386.1185560155262 )
(ra, dec): ( 71.89388932338906 ,  -37.34102758827357 )
```

In the attached notebook, we confirmed that transforming between pixel and sky coordinates gives the same result as the catalog. Here is the output of the relevant cell:

```
RA, Dec from X, Y:  (71.8938893234, -37.3410275883)
X, Y from RA, Dec:  (1730.3, 386.12)
Sky to pixel agrees!
```

Step 7 Step Execution Status: **Pass**

Description

Repeat steps 2-6, but now with difference images created by the Alert Production pipeline (for example, in the 'ap_verify' test data processing).

Expected Result

Actual Result

See the results in the attached notebook, where we successfully verified all of the PVI content for a difference image and its associated diaSrc catalog.

5.1.3.27 LVV-T141 - Verify implementation of Production Monitoring

Version **1**. Status **Approved**. Open *LVV-T141* test case in Jira.

Demonstrate monitoring capabilities that give real-time view of pipeline execution and production systems usage/load.

Preconditions:

Data set and mechanism for Production Orchestration as outlined in LVV-T140.

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Process data with the Data Release Production payload, starting from raw science images and generating science data products, placing them in the Data Backbone.

Expected Result

Actual Result
300 sq. degrees of simulated LSST data (DESC DC2) was processed as part of the DP0.2 processing campaign at the Interim Data Facility. The production orchestration software used was PanDA.

Step 2 Step Execution Status: **Pass**

Description
While DRP processing is executing, monitor the progress and resource usage of processing.

Expected Result
Ability to monitor in real-time the orchestrated production processing, including resource usage.

Actual Result
During the execution of jobs in the PanDA system, production, resources usage was monitored via the PanDA monitoring interface. DOMA_GOOGLE_LSST_Queues.png shows LSST processing jobs in in the DOMA_GOOGLE_LSST_* queues. DP02 Processing Runs.png shows a snapshot of jobs jobs in the PanDA workflow monitor during processing.

5.1.3.28 LVV-T140 - Verify implementation of Production Orchestration

Version **1**. Status **Approved**. Open *LVV-T140* test case in Jira.

Demonstrate use of orchestration software to perform batch production on LSST compute platform(s).

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify an appropriate precursor dataset.

Expected Result

Actual Result

The dataset to be used is the DP0.2 / DESC DC2 dataset at the IDF. DP0.2 Production Processing used stack v23 on IDF's PanDA system.

Step 2 Step Execution Status: **Pass**

Description

Execute a batch processing job using the orchestration system, and confirm (manually and/or via QA tools typically used for HSC reprocessing) that the pipeline executed and produced all expected products (or error logs in cases of failure).

Expected Result

Calexp single-visit and coadd images, and associated catalogs, are present in a Butler repository. Logs of the processing are available to be inspected for identification of problems in the processing.

Actual Result

Batch processing was carried in the Google Cloud environment at the IDF. The production orchestration system deployed and used at the IDF (Interim Data Facility) was PanDA. The BPS (Batch Processing Service) software (part of the LSST distribution) was used to interface between the Butler and PanDA using a set of yaml job submission files. A set of six PanDA job queues were established with different retry characteristics and memory limits. In excess of 4,000 cores were available in a number of queues and memory/core configurations within the IDF. 20K raw exposures/visits were pre-loaded into S3 storage to be processed via thePanDA system. After processing was complete, a copy of the Butler registry was made for end user access. All batch processing for DP0.2 is described in Jira ticket: PREOPS-905. Report https://rtn-039.lsst.io/ describes the full details of the DP0.2 production processing.

Logs of the processing
Google Cloud logging system was used to collect logging information output by the production pipelines and organize it into a searchable form sorted by date of production. Snapshots of logs showing the status of jobs is attached.

Show that single visit, coadds and catalogs are present in the DP0.2 Butler repository.
These data products can be accessed via the DP0.2 Butler programmatically at the IDF with the attached script. This clearly shows the availability of processed data products

5.1.3.29 LVV-T129 - Verify implementation of Provide Calibrated Photometry

Version **1**. Status **Approved**. Open *LVV-T129* test case in Jira.

Verify that the DMS provides photometry calibrated in AB mags and fluxes (in nJy) for all measured objects and sources. Must be tested for both DRP and AP products.

Preconditions:

Execution status: **Initial Pass**

Final comment:
Executed at the IDF using Science Pipelines version w_2022_32. A portion of this test case can be executed by running the script test_LVV-T129.py, which is available in the test report github repository's "scripts/" directory. Qserv tables (i.e., Object tables) were checked via the notebook test_LVV-T129.ipynb, which is available in the test report github repository's "notebooks" directory.

This Test Case is granted status of "Initial Pass" because the DIAObject table in Qserv does not clearly demonstrate that fluxes are provided in nJy. Though the fluxes are shown to be reasonable values if they are in nJy (and inspection of code could confirm this), we choose not to grant this test a full PASS until the DIAObject table is clearly reporting fluxes in nJy.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as 'DATA/path', then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

Tests were performed by executing test_LVV-T28.py, which contains the following:

# For DP0.2 data on the IDF:
config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)

Step 2 Step Execution Status: **Pass**

Description

Ingest the data products from an appropriate DRP-processed dataset.

Expected Result

Actual Result

By default, the test script contains the following detector/visit combination. This can be changed if desired.

# Select an arbitrary source catalog from a PVI and difference image:
instrument = 'LSSTCam-imSim'
detector = 5
visit = 924041

Run the test
python test_LVV-T129

Step 3 Step Execution Status: **Pass**

Description

Confirm that AB-calibrated magnitudes and fluxes are available for all measured Sources and Objects. [An enhanced verification could include matching the sources to an external source catalog and comparing the magnitudes to show that they are well-calibrated.]

Expected Result

Calibrated fluxes and magnitudes are available for all sources, as well as tools to convert measured fluxes to magnitudes (and vice-versa).

Actual Result

The test script confirms that AB-calibrated magnitudes and fluxes in nJy are available for all Sources. This is checked for both PVIs (i.e., "calexps") and difference images.The script also compares the results from the Science Pipelines to those calculated via transformations from nJy to ABmag in 'astropy' tools.

python test_LVV-T129.py
lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}

TRUE: src table fluxes are available in calibrated nJy and ABmag.
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}

TRUE: goodSeeingDiff_diaSrc table fluxes are available in calibrated nJy and ABmag.

Step 4 Step Execution Status: **Pass**

Description

Ingest the data products from an appropriate AP processing dataset.

Expected Result

Actual Result
This was confirmed in Step 3 for single-visit AP processing. In the following step, we will confirm the fluxes and magnitudes for Object tables.

Step 5 Step Execution Status: **Initial Pass**

Description
Confirm that AB-calibrated magnitudes and fluxes are available for all measured Sources, DIASources, and Objects. [An enhanced verification could include matching the sources to an external source catalog and comparing the magnitudes to show that they are well-calibrated.]

Expected Result
Calibrated fluxes and magnitudes are available for all Sources, DIASources, and Objects, as well as tools to convert measured fluxes to magnitudes (and vice-versa).

Actual Result
For the Object and DIAObject tables, we instead query the Qserv database via the Rubin Science Platform. The results are seen in notebook test_LVV-T129.ipynb.

The notebook demonstrates that the Object table (i.e., measurements from DRP processed coadd images) meets the requirement – all fluxes are in calibrated nanoJanskys. However, the DIAObject table does not report units with the flux values. Although we show in the attached notebook that these fluxes have reasonable values assuming they are in nJy, the lack of units in the database and schema leads us to characterize this step of the test as an INITIAL PASS. An issue has been created to fix this, and the test case will be executed again in the futue.

5.1.3.30 LVV-T127 - Verify implementation of Provide Source Detection Software

Version **1**. Status **Approved**. Open *LVV-T127* test case in Jira.

Verify that the DMS provides source detection software that can be applied to calibrated images, including both difference images and coadds. This will be verified using simulated data, but could also be done by inserting artificial sources into existing datasets.

Preconditions:

Execution status: **Pass**

Final comment:
See attached artifact notebook “test_LVV-T127.ipynb” for details.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Run DRP and AP processing, including source detection and measurement algorithms, on a small portion of the data from a simulated dataset.

Expected Result
Source catalogs containing measurements of all sources detected in the input images.

Actual Result
For this test, we use Data Preview 0.2 data, which is the reprocessed version of the simulated DC2 dataset. All work for this test took place in the notebook “test_LVV-T127.ipynb”, which is archived alongside the Test Report for this campaign.

In the notebook, a number of explorations show that the source detection and measurement algorithms ran, and produced reasonable measurements, as required.

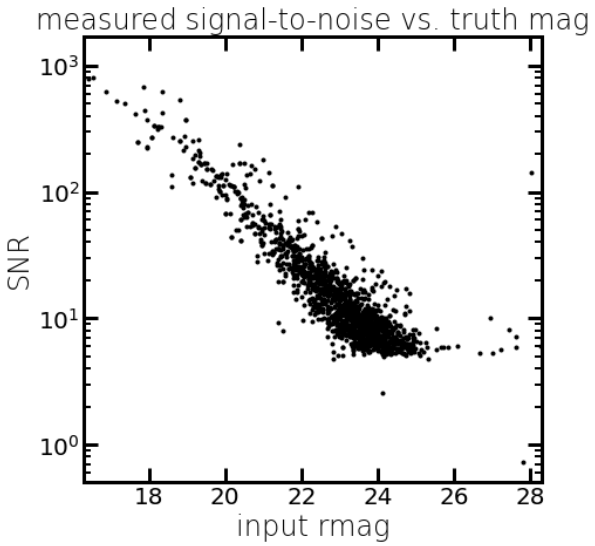
Step 2 Step Execution Status: **Pass**

Description
Confirm that the output repos contain catalogs of source detections. Compare these output catalogs to the original simulated source catalogs, and confirm that a large fraction of the sources within a reasonable signal-to-noise range were recovered.

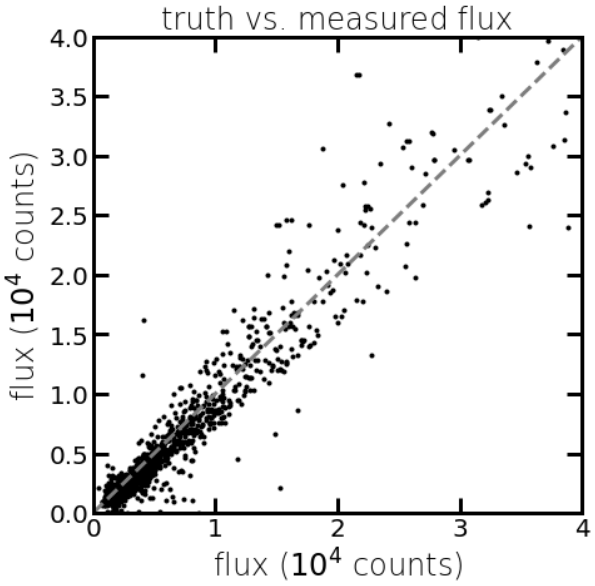
Expected Result
Most sources above a reasonable S/N threshold were detected, and their measured fluxes are reasonably close to the simulated inputs.

Actual Result

We demonstrate in the notebook (from which the figure below is derived) that sources are detected to a limit of S/N>5, which reaches a depth of at least 24th magnitude.



We also demonstrate that the measurements agree with the “true” fluxes of the simulated sources rather well:



5.1.3.31 LVV-T79 - Verify implementation of PSF-Matched Coadds

Version **1**. Status **Approved**. Open *LVV-T79* test case in Jira.

Verify that the DRP pipelines produce PSF matched coadds.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute the AssembleCoaddTask, specifying via configuration that PSF-matched coadds should be created.

Expected Result

Pipetask executes and produces coadd images.

Actual Result

Working on the USDF (SLAC) with the LSST Science Pipelines set up, we defined a brief pipeline that explicitly specifies that 'psfMatched' warps should be created. For this test, we use the 'rc2_subset' data, which has already been processed from end to end. The PSF-matched warps are created in the 'step3' subset of 'rc2_subset' processing, so we can use those to create a PSF-matched coadd.

Contents of psfmatched_coadd.yaml:

description: The DRP pipeline specialized for rc2_subset processing in jenkins and tutorials
imports:
- $DRP_PIPE_DIR/pipelines/HSC/DRP-RC2.yaml
tasks:
assemblePsfMatchedCoadd:
class: lsst.pipe.tasks.assembleCoadd.AssembleCoaddTask
config:

warpType: psfMatched

Execute the pipeline as follows:
pipetask run -b $RC2_SUBSET_DIR/SMALL_HSC/butler.yaml -d “tract = 9813 AND skymap = ‘hsc_rings_v1’ AND patch in (38, 39, 40, 41) AND band=’i’” -p $RC2__DIR/psfmatched_coadd.yaml#assemblePsfMatchedCoadd -i u/$USER/step3 -o u/$USER/psf_matched_coadds –register-dataset-types -j 4

This pipeline executed its 4 quanta in about 5 minutes.

Step 2 Step Execution Status: **Pass**

Description

Identify the path to the data repository, which we will refer to as ‘DATA/path’, then execute the following:

Example Code

```
from lsst.daf.butler import Butler
repo = 'Data/path'
collection = 'collection'
butler = Butler(repo, collections=collection)
```

Expected Result

Butler repo available for reading.

Actual Result

from lsst.daf.butler import Butler
repo = ‘/sdf/data/rubin/u/jcarlin/repos/rc2_subset/SMALL_HSC’

# Output collection from pipetask run in Step 1:
collection = ‘u/jcarlin/psf_matched_coadds’
butler = Butler(repo, collections=collection)

# Initialize a second Butler pointing to the original rc2_subset processing:
collection2 = ‘u/jcarlin/step3’
butler2 = Butler(repo, collections=collection2)

Step 3 Step Execution Status: **Pass**

Description

Verify that PSF-matched coadds were created.

Expected Result

Actual Result

In an ipython session, we executed the following commands. The object is to demonstrate (a) that the coadd was created, and (b) that the PSF-matched coadd differs from the direct coadd in the “step3” collection.

```
# Is “coadd” an Exposure object?
coadd
Out[29]: <lsst.afw.image.exposure.ExposureF at 0x7f9be81e4db0>
```

Yes, it is a non-empty exposure. Next, extract the newly-created, PSF-matched coadd and the direct-warp coadd, and compare their pixel values.

```
import numpy as np

# Get the PSF-matched coadd:
coadd = butler.get('deepCoadd', tract=9813, patch=38, band='i')

# Get the direct coadd:
coadd2 = butler2.get('deepCoadd', tract=9813, patch=38, band='i')

# Copy the PSF-matched coadd:
coadd_copy = coadd.clone()
# Subtract the direct coadd's image array from the PSF-matched image array:
coadd_copy.image.array -= coadd2.image.array

# Extract some statistics of the resulting array:
```

In [26]: np.mean(coadd_copy.image.array)
Out[26]: 1.31110555e-05

In [27]: np.median(coadd_copy.image.array)
Out[27]: 1.4901161e-08

In [28]: np.std(coadd_copy.image.array)
Out[28]: 0.0049309013

The two coadd images differ, but only by a small amount, which is verification that the PSF-matched coadd has been created successfully.

5.1.3.32 LVV-T1830 - Verify Implementation of Scientific Visualization of Camera Image Data

Version **1**. Status **Approved**. Open *LVV-T1830* test case in Jira.

Verify that all scientific visualization of camera image data uses the coordinate systems defined in LSE-349.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Identify an image containing bright saturated stars. Load this image into an image viewer such as Firefly or DS9.

Expected Result

Image with bright stars is displayed.

Actual Result

Via the RSP Portal aspect, we executed a search of the Source table over a 5-degree radius, with constraints "pixelFlags_saturated = 1," "pixelFlags_saturatedCenter = 1," and "detect_isPrimary = 1" to identify saturated stars. We then took the necessary information to create a dataId, and selected the corresponding calexp image within the Notebook aspect. The dataId and relevant commands from the notebook are:

```
from lsst.daf.butler import Butler
import lsst.afw.display as afwDisplay
from firefly_client import FireflyClient


config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = Butler(config, collections=collection)


dataId = {'instrument': 'LSSTCam-imSim', 'detector': 180, 'visit': 888370, 'exposure':888370, 'band':'g'}
calexp = butler.get('calexp', dataId=dataId)


afwDisplay.setDefaultBackend('firefly')
afw_display = afwDisplay.Display(frame=1)
afw_display.mtv(calexp)
```

This results in a calexp image displayed in Firefly.

Step 2 Step Execution Status: **Pass**

Description

Confirm that each of the following is true:

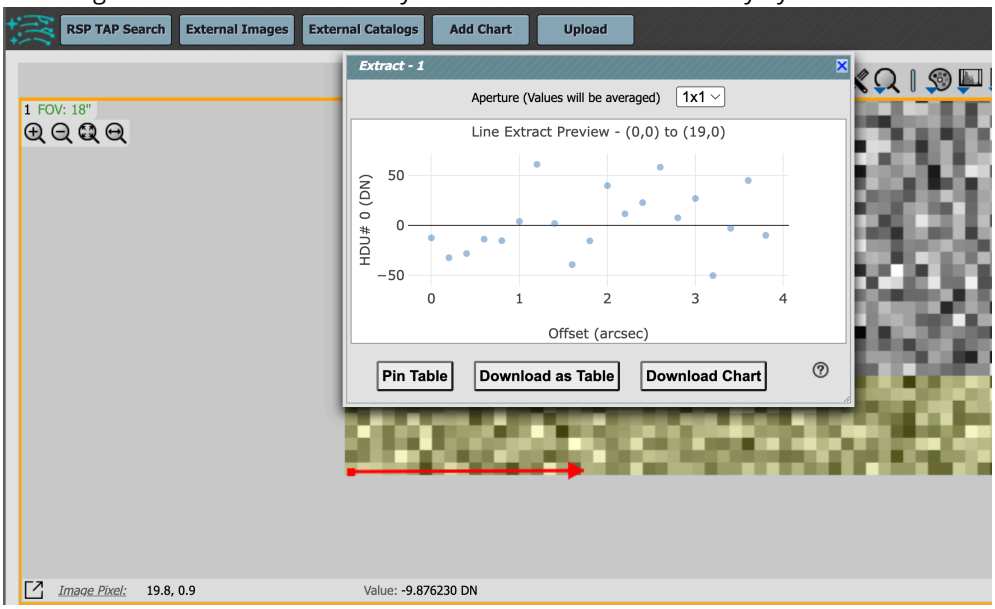
- the XY coordinate origin is at the lower left,
- the x-coordinate increases left-to-right, and the y-coordinate increases bottom-to-top
- bleed trails of saturated stars are vertical (i.e., the parallel transfer direction is oriented vertically)
- the sky orientation places east 90 degrees counter-clockwise from north

Expected Result

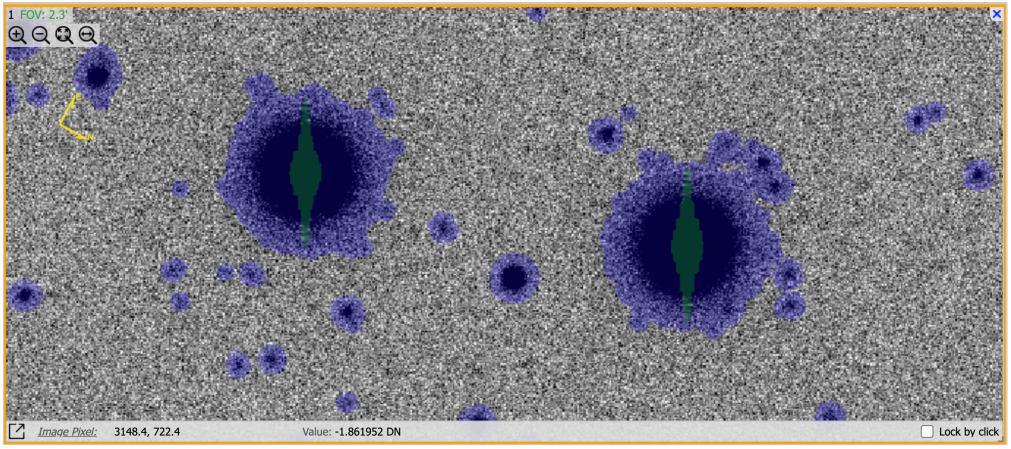
Via coordinate grid overlays or similar, an image is demonstrated to meet the necessary conditions.

Actual Result

In the first screenshot, we use the “extract line from image” tool to demonstrate the first two conditions. The red arrow selecting pixels for statistics begins at the lower left corner, and the inset plot says “Line Extract Preview - (0, 0) to (19, 0)”, thus confirming that the lower-left corner is the coordinate origin. The arrow shows directionality, so that the figure demonstrates the arrowhead is at coordinate (19, 0). This confirms that the x-coordinate increases left-to-right. We confirmed that the y-coordinate increases vertically by a similar method.



The next screenshot shows two saturated stars. The mask plane has been displayed atop the image, with pixels flagged as “SATURATED” shown in green. From the two saturated stars, it is clear that bleed trails are oriented vertically. Finally, the sky orientation is shown by the yellow arrows near the upper left, which demonstrate that east is oriented 90 degrees counter-clockwise from north, as required.



We have thus demonstrated that camera image data is visualized using the coordinates as defined in LSE-349.

5.1.3.33 LVV-T145 - Verify implementation of Task Configuration

Version **1**. Status **Approved**. Open *LVV-T145* test case in Jira.

Verify that the DMS software provides configuration control to define, override, and verify the configuration for a DMS Task.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Inspect software design to verify that one can define the configuration for a Task.

Expected Result

Actual Result

As an example, take isrTask. Inspection reveals the following, which is a portion of the configuration for IsrTask:

```
class IsrTaskConfig(pipeBase.PipelineTaskConfig,
                    pipelineConnections=IsrTaskConnections):
    """Configuration parameters for IsrTask.

    Items are grouped in the order in which they are executed by the task.
    """
    datasetType = pexConfig.Field(
        dtype=str,
        doc="Dataset type for input data; users will typically leave this alone, "
        "but camera-specific ISR tasks will override it",
        default="raw",
    )

    fallbackFilterName = pexConfig.Field(
        dtype=str,
        doc="Fallback default filter name for calibrations.",
        optional=True
    )
    useFallbackDate = pexConfig.Field(
        dtype=bool,
        doc="Pass observation date when using fallback filter.",
        default=False,
    )
    expectWcs = pexConfig.Field(
        dtype=bool,
        default=True,
        doc="Expect input science images to have a WCS (set False for e.g. spectrographs)."
    )
    fwhm = pexConfig.Field(
        dtype=float,
        doc="FWHM of PSF in arcseconds.",
        default=1.0,
    )
```

Step 2 Step Execution Status: **Pass**

Description

Run a Task with a known invalid configuration. Verify that the error is caught before the science algorithm executes.

Expected Result

Actual Result

With the science pipelines w_2022_35 set up on the USDF devl machines, execute the following:

setup -j -r /sdf/group/rubin/u/jcarlin/repos/rc2_subset/
export NUMPROC=8

Copy the default pipeline for the rc2_subset processing. We will modify this for each test step.
cp $RC2_SUBSET_DIR/pipelines/DRP.yaml ./

Edit this file to contain the (deliberately wrong) configuration:
tasks:
  characterizeImage:
    class: lsst.pipe.tasks.characterizeImage.CharacterizeImageTask
    config:
      detection.thresholdValue: 'starwars'

Execute the pipeline task for "step1":

pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p DRP.yaml#nightlyStep1 -i HSC/RC2/defaults –register-dataset-types -o u/$USER/LVV-T145_step1 -d "detector in (42) AND visit in (11690, 11698)"

pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p DRP.yaml#nightlyStep1 -i HSC/RC2/defaults –register-dataset-types -o u/$USER/LVV-T145_step1

As expected, this fails with the following error:

ERROR 2022-08-30T15:02:50.570-07:00 lsst.daf.butler.cli.utils ()(utils.py:1049) - Caught an exception, details are in traceback:
Traceback (most recent call last):
  File "/sdf/group/rubin/sw/conda/envs/lsst-scipipe-4.1.0/share/eups/Linux64/pex_config/g849534e15f+3b870f08dc/python/lsst/pe
line 801, in __set__
    self._validateValue(value)
  File "/sdf/group/rubin/sw/conda/envs/lsst-scipipe-4.1.0/share/eups/Linux64/pex_config/g849534e15f+3b870f08dc/python/lsst/pe
line 144, in _validateValue
    Field._validateValue(self, value)
  File "/sdf/group/rubin/sw/conda/envs/lsst-scipipe-4.1.0/share/eups/Linux64/pex_config/g849534e15f+3b870f08dc/python/lsst/pe
line 628, in _validateValue
    raise TypeError(msg)
TypeError: Value starwars is of incorrect type str. Expected type float

We have thus verified that an invalid configuration causes an error before the tasks execute.

Step 3 Step Execution Status: **Pass**

Description

Run a simple task with two different configurations that make a material difference for a Task. E.g., specify a different source detection threshold. Verify that the configuration is different between the two runs through

difference in recorded provenance and in results.

Expected Result

Actual Result

Now we will change the threshold from the previous step to a valid numerical value. We will then run the task with two different values for this detection threshold, and confirm that the results differ.

tasks:
characterizeImage:
class: lsst.pipe.tasks.characterizeImage.CharacterizeImageTask
config:
detection.thresholdValue: 100

pipetask –long-log run -j $NUMPROC -b ${RC2_SUBSET_DIR}/SMALL_HSC/butler.yaml -p DRP.yaml#nightlyStep1 -i HSC/RC2/defaults –register-dataset-types -o u/$USER/LVV-T145_step1_thresh100 -d “detector in (42) AND visit in (11690, 11698)”

Now change the threshold to 30, and run the pipetask again, specifying u/$USER/LVV-T145_step1_thresh30 as the output collection.

The attached script “test_LVV-T145.py” reads the ‘src’ catalog from each of the two collections in turn, and compares the number of sources in each catalog.

Executing this prints the following to the screen:
python test_LVV-T145.py

lsst_distrib g0b29ad24fb+434521fcbd current w_2022_35 setup
src1 length: 3172
src2 length: 3212
run1 threshold: 100.0
run2 threshold: 30.0

This confirms that the change in threshold affected the number of detected sources, and that the thresholds were

persisted in metadata in the butler collections.

5.1.3.34 LVV-T144 - Verify implementation of Task Specification

Version **1**. Status **Approved**. Open *LVV-T144* test case in Jira.

Verify that the DMS provides the ability to define a new or modified pipeline task without recompilation.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Inspect software architecture. Verify that there exist Tasks that can be run and configured without re-compilation.

Expected Result

Confirmation that the software architecture has allowed for reconfiguring and running Tasks without recompilation.

Actual Result

As an example, take isrTask. Inspection reveals the following, which is a portion of the configuration for IsrTask:

```python
class IsrTaskConfig(pipeBase.PipelineTaskConfig,
                    pipelineConnections=IsrTaskConnections):
    """Configuration parameters for IsrTask.

    Items are grouped in the order in which they are executed by the task.
    """
    datasetType = pexConfig.Field(
        dtype=str,
        doc="Dataset type for input data; users will typically leave this alone, "
        "but camera-specific ISR tasks will override it",
        default="raw",
    )

    fallbackFilterName = pexConfig.Field(
        dtype=str,
        doc="Fallback default filter name for calibrations.",
        optional=True
    )
    useFallbackDate = pexConfig.Field(
        dtype=bool,
        doc="Pass observation date when using fallback filter.",
        default=False,
    )
    expectWcs = pexConfig.Field(
        dtype=bool,
        default=True,
        doc="Expect input science images to have a WCS (set False for e.g. spectrographs)."
    )
    fwhm = pexConfig.Field(
        dtype=float,
        doc="FWHM of PSF in arcseconds.",
        default=1.0,
    )
```

These configuration parameters can be changed in a .yaml pipeline that is invoked at pipetask runtime.

Step 2 Step Execution Status: **Pass**

Description

Verify that tasks can consist of multiple subtasks chained together.

Expected Result

Confirmation that the software architecture has allowed for the use of subsets and chains of tasks.

Actual Result

An example of a YAML pipeline (in this case, one that calculates the crosstalk correction) is seen below. One can see that it satisfies the requirement from step 1 that the configuration parameters can be specified without recompilation of code. At the bottom, this excerpt also shows an example of subtasks being chained together to create larger tasks.

```
description: cp_pipe CROSSTALK calibration construction.
tasks:
  isr:
    class: lsst.ip.isr.IsrTask
    config:
      connections.ccdExposure: 'raw'
      connections.outputExposure: 'cpCrosstalkProc'
      doWrite: true
      doOverscan: true
      doAssembleCcd: true
      doBias: true
      doVariance: false
      doLinearize: true
      doCrosstalk: false
      doBrighterFatter: false
      doDark: false
      doStrayLight: false
      doFlat: false
      doFringe: false
      doApplyGains: false
      doDefect: true
      doSaturationInterpolation: false
      growSaturationFootprintSize: 0
  crosstalkExtract:
    class: lsst.cp.pipe.measureCrosstalk.CrosstalkExtractTask
    config:
      connections.inputExp: 'cpCrosstalkProc'
      connections.outputRatios: 'cpCrosstalkRatio'
  crosstalkSolve:
    class: lsst.cp.pipe.measureCrosstalk.CrosstalkSolveTask
    config:
      connections.inputRatios: 'cpCrosstalkRatio'
      connections.outputCrosstalk: 'crosstalk'
subsets:
  crosstalk:
    subset:
      - crosstalkExtract
      - crosstalkSolve
```

Step 3 Step Execution Status: **Pass**

Description

Verify that an example science algorithm can be run through one of these Tasks.

Expected Result

Successful Task execution with different configurations, including confirmation that the outputs are different from tasks with altered configurations.

Actual Result

For this, we refer to verification performed in LVV-T145, where we demonstrated that configurations can be changed in YAML specification files, and that these changes can be demonstrated to take effect.

Executing the script from LVV-T145 prints the following to the screen:
python test_LVV-T145.py

lsst_distrib g0b29ad24fb+434521fcbd current w_2022_35 setup
src1 length: 3172
src2 length: 3212
run1 threshold: 100.0
run2 threshold: 30.0

...where run1 and run2 were source detection runs using detection thresholds of 100 and 30.

This confirms that the change in threshold affected the number of detected sources, and that the thresholds were persisted in metadata in the butler collections.

5.1.3.35 LVV-T74 - Verify implementation of Template Coadds

Version **1**. Status **Approved**. Open *LVV-T74* test case in Jira.

Verify that the DMS can produce Template Coadds for DIA processing.

Preconditions:

Execution status: **Pass**

Final comment:

See attached artifact notebook “test_LVV-T74.ipynb” for details.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Perform the steps of Alert Production (including, but not necessarily limited to, single frame processing, ISR, source detection/measurement, PSF estimation, photometric and astrometric calibration, difference imaging, DIASource detection/measurement, source association). During Operations, it is presumed that these are automated for a given dataset.

Expected Result

An output dataset including difference images and DIASource and DIAObject measurements.

Actual Result

Logged into the RSP at the Interim Data Facility, and accessed the shared butler repositories containing Data Preview 0.2 data. All work for this Test Case is in Jupyter notebook “test_LVV-T74.ipynb”, which is archived along with the test report.

Step 2 Step Execution Status: **Pass**

Description

Verify that the expected data products have been produced, and that catalogs contain reasonable values for measured quantities of interest.

Expected Result

Actual Result

We confirmed that the expected data products are present, including DIASource tables, DIAObject measurements and their associations with DIASource, template images, and that all have been photometrically and astrometrically calibrated.

Step 3 Step Execution Status: **Pass**

Description

Confirm that the template coadds have been created and are well-formed.

Expected Result

Actual Result

Finally, we displayed a template coadd and compared it to the original image. See notebook “test_LVV-T74.ipynb” for details.

A Documentation

The verification process is defined in LSE-160. The use of Docsteady to format Jira information in various test and planing documents is described in DMTN-140 and practical commands are given in DMTN-178.

B Acronyms used in this document

Acronym	Description
ADQL	Astronomical Data Query Language
AP	Alert Production
AT	Auxiliary Telescope
AURA	Association of Universities for Research in Astronomy
BNL	Brookhaven National Laboratory
BPS	Batch Production Service
CCD	Charge-Coupled Device
CR	Change Request
CSC	Commandable SAL Component
CSV	Comma Separated Values
CTIO	Cerro Tololo Inter-American Observatory
DAC	Data Access Center
DC2	Data Challenge 2 (DESC)
DEC	Declination
DES	Dark Energy Survey
DESC	Dark Energy Science Collaboration
DIA	Difference Image Analysis
DM	Data Management
DMS	Data Management Subsystem
DMSR	DM System Requirements; LSE-61
DMTN	DM Technical Note
DMTR	DM Test Report
DOE	Department of Energy
DP0	Data Preview 0
DRP	Data Release Production

DS9	Deep Space 9 (specific astronomical data visualisation application; SAOImage)
FITS	Flexible Image Transport System
HD	historical data
HSC	Hyper Suprime-Cam
IDF	Interim Data Facility
IPAC	No longer an acronym; science and data center at Caltech
ISR	Instrument Signal Removal
JSON	JavaScript Object Notation
LATISS	LSST Atmospheric Transmission Imager and Slitless Spectrograph
LDM	LSST Data Management (Document Handle)
LLNL	Lawrence Livermore National Laboratory
LSE	LSST Systems Engineering (Document Handle)
LSP	LSST Science Platform (now Rubin Science Platform)
LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
LVV	LSST Verification and Validation
MJD	Modified Julian Date (to be avoided; see also JD)
NOIRLab	NSF's National Optical-Infrared Astronomy Research Laboratory; `https://nationalastro.org`
OBS	Organisation Breakdown Structure
PDR	Preliminary Design Review
PMCS	Project Management Controls System
PSF	Point Spread Function
PVI	Processed Visit Image
PanDA	Production ANd Distributed Analysis system
QA	Quality Assurance
RA	Right Ascension
RDO	Rubin Directors Office
RDP	Rubin Data Production
RMS	Root-Mean-Square
RSP	Rubin Science Platform
S3	(Amazon) Simple Storage Service
SDSS	Sloan Digital Sky Survey

SITCOM	System Integration, Test and Commissioning
SLAC	SLAC National Accelerator Laboratory
SOAR	Southern Astrophysical Research Telescope
TAI	International Atomic Time
TAP	Table Access Protocol
TOPCAT	Tool for OPerations on Catalogues And Tables
UI	User Interface
UK	United Kingdom
US	United States
USDF	United States Data Facility
UT1	Universal Time 1
WCS	World Coordinate System
YAML	Yet Another Markup Language
arcsec	arcsecond second of arc (unit of angle)
deg	degree; unit of angle