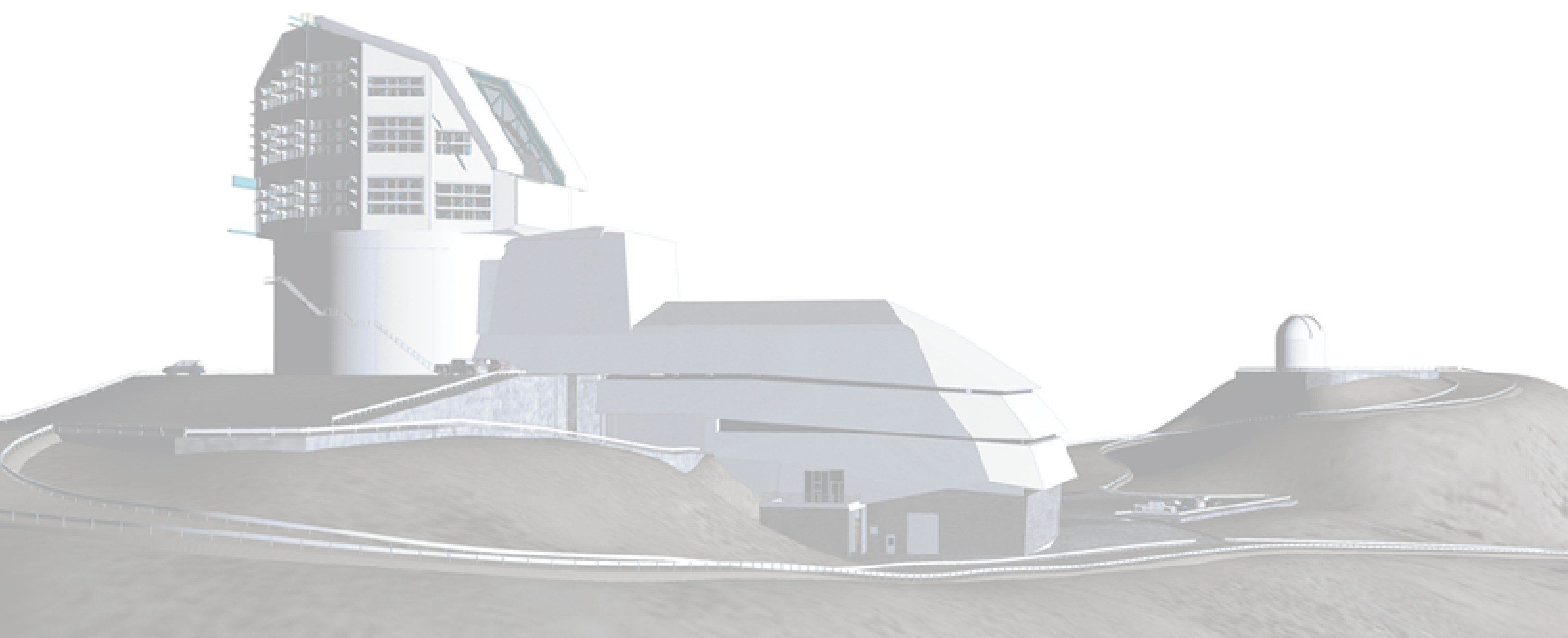
Vera C. Rubin Observatory
Data Management

LVV-P106: Data Management Acceptance Test Campaign, Fall 2023 Test Plan and Report

Jeffrey Carlin

DMTR-401

Latest Revision: 2024-05-16



Abstract

This is the test plan and report for **Data Management Acceptance Test Campaign, Fall 2023**, an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2024-05-16 . The most recent change to the document repository was on 2024-05-16.

Change Record

Version	Date	Description	Owner name
	2023-07-01	First draft	Jeffrey Carlin
1.0	2024-05-08	Test campaign LVV-P106 completed and results approved. DM-40311	Jeffrey Carlin

Document curator: Jeffrey Carlin
Document source location: `https://github.com/lsst-dm/DMTR-401`
Version from source repository: 52e27ab

Contents

LVV-P106: Data Management Acceptance Test Campaign, Fall 2023 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, and camera test-stand data, will be used when appropriate.

1.3 Document Overview

This document was generated from Jira, obtaining the relevant information from the LVV-P106 Jira Test Plan and related Test Cycles (LVV-C260).

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 2, 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-P106 in Jira is **Completed** .

1.4 References

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

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

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

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

[5] **[LDM-503]**, O'Mullane, W., Swinbank, J., Juric, M., et al., 2023, Data Management Test Plan, URL `https://ldm-503.lsst.io/`, Vera C. Rubin Observatory Data Management Controlled Document LDM-503

[6] **[LSE-160]**, Selvy, B., 2013, Verification and Validation Process, URL `https://ls.st/LSE-160`, Vera C. Rubin Observatory 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) and the development cluster at the USDF. In particular, we will use version 26 of the Pipelines for most tests; some tests will use more recent weekly builds of the Pipelines.

2.3 Related Documentation

No additional documentation provided.

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-P106 owner:	**Jeffrey Carlin**	
	T. Cycle LVV-C260 owner:	**Jeffrey Carlin**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T146	Leanne Guy	Leanne Guy	
LVV-T1240	Jim Bosch	Jeffrey Carlin	
LVV-T132	Leanne Guy	Jeffrey Carlin	
LVV-T62	Jim Bosch	Jeffrey Carlin	
LVV-T41	Jim Bosch	Jeffrey Carlin	
LVV-T97	Kian-Tat Lim	Jeffrey Carlin	
LVV-T1946	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1947	Jeffrey Carlin	Jeffrey Carlin	
LVV-T28	Colin Slater	Jeffrey Carlin	
LVV-T142	Colin Slater	Jeffrey Carlin	
LVV-T1748	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1759	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1758	Jeffrey Carlin	Jeffrey Carlin	
LVV-T149	Leanne Guy	Jeffrey Carlin	
LVV-T40	Jeffrey Carlin	Jeffrey Carlin	

4 Test Campaign Overview

4.1 Summary

T. Plan LVV-P106:		Data Management Acceptance Test Campaign, Fall 2023		Completed
T. Cycle LVV-C260:		Data Management Acceptance Test Campaign, Fall 2023		Done
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T146	1	Pass		
LVV-T1240	1	Pass	Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF. The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T40_T1240.ipynb".	
LVV-T132	1	Pass		
LVV-T62	2	Pass	Test executed with science pipelines version w_2023_34 in the RSP Notebook aspect at the USDF. The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T62.ipynb".	
LVV-T41	1	Pass	Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF. The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T41.ipynb".	

LVV-T97	1	Pass	Executed at the USDF using the w_2023_43 version of the Science Pipelines.
			
			In addition to demonstrating that changing the "RELEASE_ID" results in unique IDs, we further examine the code itself to demonstrate that the uniqueness of IDs is ensured by the way the code is implemented.
			
			The implementation of how the 64 bits of the Source/Object IDs are apportioned is in the `_IdGeneratorBits` class, and especially its `__post_init__`:
			
			• it gets the maximum number of bits needed to pack the data ID (based on `{visit, detector}` for Source, and `{tract, patch}` for Object) and turns that into the number of distinct data IDs it could hold (`n_data_ids`);
			• it multiplies that with the `n_releases` option to identify how many "upper" bits need to be reserved for `n_data_ids*n_releases` "catalog_ids", and sets `n_counters` to be the number of values remaining in the Source or Object 64-bit ID to count sources or objects within one image.
			
			Because some interfaces historically wanted to count bits rather than just multiply integers, there is a mix of direct multiplication and log2 addition. But you can see the result most clearly in `FullIdGenerator.arange` and `FullIdGenerator.catalog_id`; a full Source or Object ID is:
			
			`id = counter + n_counters * catalog_id`
			
			or
			
			`id = counter + n_counters * (packed_data_id + n_data_ids * release_id)`
			
			and hence the releases will get distinct IDs as long as counter values are less than `n_counters` and the packed data IDs are less than `n_data_ids`. (And there are several checks throughout the file for those criteria).

LVV-T1946	1	Pass	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. The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.
LVV-T1947	1	Pass	This test case can be executed by running the scripts test_LVV-T1947_DiaSource.py, test_LVV-T1947_forcedSourceOnDiaObject.py, and test_LVV-T1947_DiaObject.py, which are available in the test report github repository's "scripts/" directory. The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.
LVV-T28	1	Pass	This test case can be executed by running the scripts test_LVV-T28.py, test_LVV-T28_forcedSource.py, and test_LVV-T28_DC2.py, which are available in the test report github repository's "scripts/" directory. The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.
LVV-T142	1	Pass	Executed at the USDF using the w_2023_43 version of the Science Pipelines.
LVV-T1748	1	Pass	
LVV-T1759	1	Pass	This test used a modified version of the analysis_tools pipeline "matchedVisitQualityCore.yaml."
LVV-T1758	1	Pass	Note that because we do not have access to u-band data, this test was performed for only y- and z-band. The steps would be unchanged for u-band data.
LVV-T149	1	Pass	Executed using the IDF Notebook, Portal, and API aspects. For the notebook execution, we used science pipelines version w_2023_34.

LVV-T40	1	Pass	Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF. The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T40_T1240.ipynb".

Table 2: Test Campaign Summary

4.2 Overall Assessment

In this test campaign, we have successfully verified 9 unique requirements from LSE-61 via the execution of 15 Test Cases (all of which passed). Of these requirements, 6 are of priority 1a, and 3 are priority 1b. The set of requirements tested in this campaign mostly cover aspects of the DM system related to the software pipelines and the facilities for data processing and handling. These tests were performed at the US Data Facility (USDF) using precursor HSC and Auxtel data.

4.3 Recommended Improvements

Not yet available.

5 Detailed Test Results

5.1 Test Cycle LVV-C260

Open test cycle *Data Management Acceptance Test Campaign, Fall 2023* in Jira.

Test Cycle name: Data Management Acceptance Test Campaign, Fall 2023
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, Auxtel data, and other data products housed at the U.S. Data Facility (USDF).

5.1.1 Software Version/Baseline

Primarily using Science Pipelines version 26 at the USDF.

5.1.2 Configuration

Not provided.

5.1.3 Test Cases in LVV-C260 Test Cycle

5.1.3.1 LVV-T146 - Verify implementation of DMS Initialization Component

Version **1**. Status **Approved**. Open *LVV-T146* test case in Jira.

Demonstrate that all components of the DM system have a defined deployment configuration within the DM deployment strategy

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Inspect each service component to check if it has a deployment configuration defined

Expected Result

All systems have a defined deployment configuration as part of the DM deployment strategy

Actual Result

Service	Location	Deployment Strategy
Archiving	Summit, USDF	S3 on Kubernetes. Summit currently ssh/NFS to bare m
Planned Observation Publication (a.k.a. ObsLocTAP)	USDF	Safir/Phalanx on Kubernetes
Prompt Processing Ingest	USDF	Kubernetes
Observatory Operations Data	Summit	Kubernetes
Observatory Control System (OCS) Controlled Pipeline	Summit	Kubernetes
Telemetry Gateway	Summit	Sasquatch/Phalanx/Kafka on Kubernetes
Prompt Processing	USDF	KNative on Kubernetes
Alert Distribution	USDF	Kubernetes
Prompt Quality Control	USDF	Part of Prompt Processing payload, with publication to S
Batch Production	USDF	The batch system is independent. It uses Slurm not on K
Offline QC	USDF	Part of Batch Production payloads, with publication to S
Bulk Distribution	USDF	Rucio/FTS3 on K8s
Data Backbone	USDF	Rucio/FTS3 + ingested on K8s
RSP Portal	Summit, USDF	Phalanx on K8s
RSP Notebook	Summit, USDF	Phalanx on K8s
RSP Web API	Summit, USDF	Phalanx on K8s

5.1.3.2 LVV-T1240 - Verify implementation of minimum astrometric standards per CCD

Version **1**. Status **Approved**. Open *LVV-T1240* test case in Jira.

Verify that each CCD in a processed dataset had its astrometric solution determined by at least **astrometricMinStandards = 5** astrometric standards.

Preconditions:

Execution status: **Pass**

Final comment:
Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF.

The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T40_T1240.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify an appropriate processed dataset for this test.

Expected Result
A dataset with Processed Visit Images.

Actual Result
For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_32 pipelines.

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 access is as follows:
repo = '/repo/main'
rc2_collection = 'HSC/runs/RC2/w_2023_32/DM-40356'
butler = Butler(repo, collections=rc2_collection)

Step 3 Step Execution Status: **Pass**

Description

Select a single visit from the dataset, and extract its calibration data. For a subset of CCDs, check how many astrometric standards contributed to the solution. Confirm that this number is at least **astrometricMinStandards = 5.**

Expected Result

At least **astrometricMinStandards** from each CCD were used in determining the WCS solution.

Actual Result

This was done 500 times, extracting the source list and WCS for 500 randomly-selected CCD/visit combinations from the repository.

It was confirmed that all CCDs selected had more than astrometricMinStandards=5 standards used in their WCS solutions. This was done using the following code to extract the number of astrometric standards for each image:

```
astrom_selection = np.where(src['calib_astrometry_used'] == True)
num_calib_astrom.append(np.size(astrom_selection))
```

In the end, we calculate the fraction of fields that met this requirement, using:
wcsFlagsPercent = (np.size(np.where(num_calib_astrom > 5))/np.size(num_calib_astrom))*100.0*u.percent

The result (from the notebook) is:
Percentage of fields with > astrometricMinStandards=5: 100.0 % – True

5.1.3.3 LVV-T132 - Verify implementation of Pre-cursor and Real Data

Version **1**. Status **Approved**. Open *LVV-T132* test case in Jira.

Demonstrate that pixel-oriented data from astronomical imaging cameras (precursor or otherwise) can be processed using LSST Science Algorithms and organized for access through the Data Butler Access Client.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description
Confirm that the CI jobs used to test DRP processing successfully run. These jobs use precursor datasets from cameras other than LSST.

Expected Result

Actual Result

Because the outputs from CI jobs are not persisted, we instead use the regularly-reprocessed HSC RC2 data. Specifically, we used the output repo from HSC-RC2 data processing, as executed using a recent weekly pipelines release (w_2023_39). The output repo is tagged with the Jira ticket number DM-40985.

The butler collection within /repo/main at the USDF is "HSC/runs/RC2/w_2023_39/DM-40985".

Step 2	Step Execution Status: **Pass**

Description

For the precursor dataset, instantiate the Butler, load the data products, and confirm that they exist as expected.

Expected Result

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

Actual Result

The Test Cases that were executed on this dataset for this test campaign demonstrate that this requirement is satisfied. Specifically, the following data products from HSC RC2 were used (among others) in the Test Cases from this campaign:

LVV-T28: 'src' and 'sourceTable'
LVV-T62: 'deepCoadd_calexp' and associated PSF
LVV-T41: 'calexp' (calibrated PVI) and associated PSF
LVV-T40, LVV-T1240: 'calexp,' 'calexp_photoCalib,' and associated WCS
LVV-T43: 'calexpBackground'
LVV-T1946: 'deepCoadd_forced_src' and 'objectTable'
LVV-T1947: 'goodSeeingDiff_diaSrc', 'forced_src_diaObject', 'diaSourceTable', and 'diaObjectTable_tract'

5.1.3.4 LVV-T62 - Verify implementation of Provide PSF for Coadded Images

Version **2**. Status **Approved**. Open *LVV-T62* test case in Jira.

Verify that all coadd images produced by the DRP pipelines include a model from which an image of the PSF at any point on the coadd can be obtained.

Preconditions:
Fully covered by preconditions for LVV-T16.

Execution status: **Pass**

Final comment:
Test executed with science pipelines version w_2023_34 in the RSP Notebook aspect at the USDF.

The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T62.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify a repo containing RC2 data with coadded images in multiple filters.

Expected Result
Multi-band data that has been processed through the coaddition stage.

Actual Result
For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_32 pipelines.

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 access is as follows:
repo = ‘/repo/main’
rc2_collection = ‘HSC/runs/RC2/w_2023_32/DM-40356’
butler = Butler(repo, collections=rc2_collection)

Step 3 Step Execution Status: **Pass**

Description

Load the exposures, then select Objects classified as point sources on at least 10 different coadd images (including all bands). Evaluate the PSF model at the positions of these Objects, and verify that subtracting a scaled version of the PSF model from the processed visit image yields residuals consistent with pure noise.

Expected Result

Images with the PSF model subtracted, leaving only residuals that are consistent with being noise.

Actual Result

Coadd tract/patch combinations were selected at random and the corresponding dataIds (datarefs) created.

To extract the PSF, the following line was executed for each dataId:
psf = c.getPsf()
...where “c” is a coadd image
that has been retrieved from the butler.

A number of these were displayed at random X, Y positions on the images to confirm that they are well-formed and retrievable at any arbitrary position.

The PSF model was then formed into an image and subtracted from selected stars to confirm that the remaining image is consistent with noise (i.e., the PSF is a good match to the stars in the image).

Finally, a larger subset of dataIds was selected, from which we test whether all calexps have an associated PSF model. The result of this test, seen in the test notebook, is as follows:

```
All patches have an associated PSF model:  True
```

If any of the 100 randomly selected patches had a malformed (or non-existent) PSF 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_LVVT62.ipynb".

5.1.3.5 LVV-T41 - Verify implementation of Generate PSF for Visit Images

Version **1**. Status **Approved**. Open *LVV-T41* test case in Jira.

Verify that Processed Visit Images produced by the DRP and AP pipelines are associated with a model from which one can obtain an image of the PSF given a point on the image.

Preconditions:

Execution status: **Pass**

Final comment:
Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF.

The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T41.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify a repo containing data with processed visit images in multiple filters.

Expected Result

Actual Result

For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_32 pipelines.

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 access is as follows:
repo = '/repo/main'
rc2_collection = 'HSC/runs/RC2/w_2023_32/DM-40356'
butler = Butler(repo, collections=rc2_collection)

Step 3 Step Execution Status: **Pass**

Description

Select Objects classified as point sources on at least 10 different processed visit images (including all bands). Evaluate the PSF model at the positions of these Objects, and verify that subtracting a scaled version of the PSF model from the processed visit image yields residuals consistent with pure noise.

Expected Result

Images with the PSF model subtracted, leaving only residuals that are consistent with being noise.

Actual Result

CCD visit/detector combinations were selected at random and the corresponding dataIds (datarefs) created.

To extract the PSF, the following line was executed for each dataId:
fvs.getPsf()
...where "fvs" is the "finalVisitSummary" that has been retrieved from the butler.

A number of these were displayed at random X, Y positions on the images to confirm that they are well-formed and retrievable at any arbitrary position.

The PSF model was then formed into an image and subtracted from selected stars to confirm that the remaining image is consistent with noise (i.e., the PSF is a good match to the stars in the image).

Finally, a larger subset of dataIds was selected, from which we test whether all calexps have an associated PSF model. The result of this test, seen in the test notebook, is as follows:

```
All CCDs have an associated PSF model:  True
```

If any of the 1000 randomly selected CCDs had a malformed (or non-existent) PSF 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_LVVT41.ipynb".

5.1.3.6 LVV-T97 - Verify implementation of Uniqueness of IDs Across Data Releases

Version **1**. Status **Approved**. Open *LVV-T97* test case in Jira.

Verify that the IDs of Objects, Sources, DIAObjects, and DIASources from different Data Releases are unique.

Preconditions:

Execution status: **Pass**

Final comment:
Executed at the USDF using the w_2023_43 version of the Science Pipelines.

In addition to demonstrating that changing the "RELEASE_ID" results in unique IDs, we further examine the code itself to demonstrate that the uniqueness of IDs is ensured by the way the code is implemented.

The implementation of how the 64 bits of the Source/Object IDs are apportioned is in the `_IdGeneratorBits` class, and especially its `__post_init__`:

- it gets the maximum number of bits needed to pack the data ID (based on `{visit, detector}` for Source, and `{tract, patch}` for Object) and turns that into the number of distinct data IDs it could hold (`n_data_ids`);
- it multiplies that with the `n_releases` option to identify how many "upper" bits need to be reserved for `n_data_ids*n_releases` "catalog_ids", and sets `n_counters` to be the number of values remaining in the Source or Object 64-bit ID to count sources or objects within one image.

Because some interfaces historically wanted to count bits rather than just multiply integers, there is a mix of direct multiplication and log2 addition. But you can see the result most clearly in `FullIdGenerator.arange` and `FullIdGenerator.catalog_id`; a full Source or Object ID is:

```
id = counter + n_counters * catalog_id
```

or

```
id = counter + n_counters * (packed_data_id + n_data_ids * release_id)
```

and hence the releases will get distinct IDs as long as counter values are less than `n_counters` and the packed data IDs are less than `n_data_ids`. (And there are several checks throughout the file for those criteria).

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify an appropriate precursor dataset to be processed through Data Release Production.

Expected Result

Actual Result

This test used the 'rc2_subset' data, which is a small set of Subaru+HSC data.

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 means of ensuring unique IDs between runs is contained in the following code from the meas_base package:

DEFAULT_RELEASE_ID = 0
"""Default release ID to embed in catalog IDs.

This can be changed globally to avoid having to override individual task
configs to set the release ID.
"""

DEFAULT_N_RELEASES = 1 # 1 means don't reserve space for releases.
"""Default number of releases to reserve space for in catalog IDs."""

For this test, we cloned meas_base and manually altered DEFAULT_RELEASE_ID and DEFAULT_N_RELEASES before each of two processing runs of rc2_subset.
git checkout -b u/jcarlin/test_id_generator

Set the following in line 44 of python/lsst/meas/base/_id_generator.py:
DEFAULT_RELEASE_ID = 8
and
DEFAULT_N_RELEASES = 10

Run scons, then 'setup -k -r .' in meas_base to set up the local version.

Working in directory "/sdf/home/j/jcarlin/u/SVV/fall2023_ATC/LVV-T97" at the USDF.

Submit the shell script that runs the standard rc2_subset procssing:
bash submit_rc2_subset.sh 2>&1 | tee rc2_subset_v8.log
(where the "8" denotes the run with DEFAULT_RELEASE_ID=8)

Edit meas_base to read "DEFAULT_RELEASE_ID = 9", then rerun it:
bash submit_rc2_subset.sh 2>&1 | tee rc2_subset_v9.log

Both processing runs successfully executed.

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

In the notebook (test_LVV-T97.ipynb) that is attached to this document's repository, we executed the following to read in the two butler repositories:

repo = '/repo/main'
rc2_subset_collection_id8 = 'u/jcarlin/lvv-t97_8'
rc2_subset_collection_id9 = 'u/jcarlin/lvv-t97_9'

# Initialize the butler repos:
butler8 = Butler(repo, collections=rc2_subset_collection_id8)
butler9 = Butler(repo, collections=rc2_subset_collection_id9)

Step 4 Step Execution Status: **Pass**

Description

After running the DRP payload multiple times, load the resulting data products (both data release and prompt products) using the Butler.

Expected Result

Multiple datasets resulting from processing of the same input data.

Actual Result

The two runs of rc2_subset processing produced Source and Object tables, which were read into the attached notebook.

Note that we did not extend the test to DiaSource or DiaObject tables, as rc2_subset is not suitable for difference imaging. Nonetheless, since difference image processing would use the same _id_generator.py code as is used here, we consider this demonstration sufficient to show that it would also for for DIA processing.

Step 5	Step Execution Status: **Pass**

Description

Inspect the IDs in the multiple data products and confirm that all IDs are unique.

Expected Result

No IDs are repeated between multiple processings of the identical input dataset.

Actual Result

In the attached notebook, we directly compared the sourceIds from sourceTable_visit and the objectIds from objectTable_tract tables.

ind_src8 = src_8.index
ind_src9 = src_9.index
np.sum(ind_src8.isin(ind_src9))

(and a similar process for the object table)

The result printed “0” to the screen, meaning that there are no IDs in common between the two runs. We have thus demonstrated that IDs are unique (or can be made to be unique) across data releases.

5.1.3.7 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:
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.

The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.

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 using "test_LVV-T1946.py," which contains the following:

# For RC2 data on the USDF:
config = '/repo/main'
collection = 'HSC/runs/RC2/w_2023_39/DM-40985'
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 selects the following instrument/detector/visit combination. This can be changed if desired.

# Select an arbitrary source catalog from a deepCoadd:
instrument = 'HSC'
band = 'i'
tract = 9615
patch = 13
skymap = 'hsc_rings_v1'

# Run the test
LVVT1946(instrument, tract, patch, band, skymap)

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

The script outputs the following, confirming that all "instFlux" values in the "deepCoadd_forced_source" table are in units of "counts", and all calibrated fluxes from the "objectTable" are in units of nJy within a reasonable range of flux values:

lsst_distrib g4213664e8e+b08e1c1b0b w_latest w_2024_04 current setup
Input dataId: {'instrument': 'HSC', 'tract': 9615, 'band': 'i', 'patch': 13, 'skymap': 'hsc_rings_v1'}
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

———————————————————-

Checking flux values for visit 9615

flux column	% of bad flux values
g_psfFlux	0.0 %
g_free_psfFlux	0.0 %
g_gaapPsfFlux	0.0 %
g_gaap0p5Flux	0.0 %
g_gaap0p7Flux	0.0 %
g_gaap1p0Flux	0.0 %
g_gaap1p5Flux	0.0 %
g_gaap2p5Flux	0.0 %
g_gaap3p0Flux	0.0 %
g_gaapOptimalFlux	0.0 %
g_kronFlux	0.0 %
g_calibFlux	0.0 %
g_ap03Flux	0.0 %
g_ap06Flux	0.0 %
g_ap09Flux	0.0 %
g_ap12Flux	0.0 %
g_ap17Flux	0.0 %
g_ap25Flux	0.0 %
g_ap35Flux	0.0 %
g_ap50Flux	0.0 %
g_ap70Flux	0.0 %
g_cModelFlux	0.0 %
g_cModelFlux_inner	0.0 %
g_free_cModelFlux	0.0 %
g_free_cModelFlux_inner	0.0 %

flux column	% of bad flux values
r_psfFlux	0.0 %
r_free_psfFlux	0.0 %
r_gaapPsfFlux	0.0 %
r_gaap0p5Flux	0.0 %
r_gaap0p7Flux	0.0 %
r_gaap1p0Flux	0.0 %
r_gaap1p5Flux	0.0 %
r_gaap2p5Flux	0.0 %

r_gaap3p0Flux 0.0 %
r_gaapOptimalFlux 0.0 %
r_kronFlux 0.0 %
r_calibFlux 0.0 %
r_ap03Flux 0.0 %
r_ap06Flux 0.0 %
r_ap09Flux 0.0 %
r_ap12Flux 0.0 %
r_ap17Flux 0.0 %
r_ap25Flux 0.0 %
r_ap35Flux 0.0 %
r_ap50Flux 0.0 %
r_ap70Flux 0.0 %
r_cModelFlux 0.0 %
r_cModelFlux_inner 0.0 %
r_free_cModelFlux 0.0 %
r_free_cModelFlux_inner 0.0 %

flux column % of bad flux values

i_psfFlux 0.0 %
i_free_psfFlux 0.0 %
i_gaapPsfFlux 0.0 %
i_gaap0p5Flux 0.0 %
i_gaap0p7Flux 0.0 %
i_gaap1p0Flux 0.0 %
i_gaap1p5Flux 0.0 %
i_gaap2p5Flux 0.0 %
i_gaap3p0Flux 0.0 %
i_gaapOptimalFlux 0.0 %
i_kronFlux 0.0 %
i_calibFlux 0.0 %
i_ap03Flux 0.0 %
i_ap06Flux 0.0 %
i_ap09Flux 0.0 %
i_ap12Flux 0.0 %
i_ap17Flux 0.0 %
i_ap25Flux 0.0 %
i_ap35Flux 0.0 %
i_ap50Flux 0.0 %
i_ap70Flux 0.0 %
i_cModelFlux 0.0 %

i_cModelFlux_inner 0.0 %
i_free_cModelFlux 0.0 %
i_free_cModelFlux_inner 0.0 %

flux column % of bad flux values

z_psfFlux 0.0 %
z_free_psfFlux 0.0 %
z_gaapPsfFlux 0.0 %
z_gaap0p5Flux 0.0 %
z_gaap0p7Flux 0.0 %
z_gaap1p0Flux 0.0 %
z_gaap1p5Flux 0.0 %
z_gaap2p5Flux 0.0 %
z_gaap3p0Flux 0.0 %
z_gaapOptimalFlux 0.0 %
z_kronFlux 0.0 %
z_calibFlux 0.0 %
z_ap03Flux 0.0 %
z_ap06Flux 0.0 %
z_ap09Flux 0.0 %
z_ap12Flux 0.0 %
z_ap17Flux 0.0 %
z_ap25Flux 0.0 %
z_ap35Flux 0.0 %
z_ap50Flux 0.0 %
z_ap70Flux 0.0 %
z_cModelFlux 0.0 %
z_cModelFlux_inner 0.0 %
z_free_cModelFlux 0.0 %
z_free_cModelFlux_inner 0.0 %

flux column % of bad flux values

y_psfFlux 0.0 %
y_free_psfFlux 0.0 %
y_gaapPsfFlux 0.0 %
y_gaap0p5Flux 0.0 %
y_gaap0p7Flux 0.0 %
y_gaap1p0Flux 0.0 %

y_gaap1p5Flux	0.0 %
y_gaap2p5Flux	0.0 %
y_gaap3p0Flux	0.0 %
y_gaapOptimalFlux	0.0 %
y_kronFlux	0.0 %
y_calibFlux	0.0 %
y_ap03Flux	0.0 %
y_ap06Flux	0.0 %
y_ap09Flux	0.0 %
y_ap12Flux	0.0 %
y_ap17Flux	0.0 %
y_ap25Flux	0.0 %
y_ap35Flux	0.0 %
y_ap50Flux	0.0 %
y_ap70Flux	0.0 %
y_cModelFlux	0.0 %
y_cModelFlux_inner	0.0 %
y_free_cModelFlux	0.0 %
y_free_cModelFlux_inner	0.0 %

5.1.3.8 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:
This test case can be executed by running the scripts test_LVV-T1947_DiaSource.py, test_LVV-

T1947_forcedSourceOnDiaObject.py, and test_LVV-T1947_DiaObject.py, which are available in the test report github repository's "scripts/" directory.

The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.

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

The test scripts contain the following:

# For RC2 data on the USDF:
config = '/repo/main'
collection = 'HSC/runs/RC2/w_2023_39/DM-40985'
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
Relevant lines from “test_LVV-T1947_DiaSource.py”:
# Select an arbitrary source catalog from a difference image:
instrument = ‘HSC’
detector = 40
visit = 36180

# Run the test
LVVT1947(instrument, visit, detector)

Relevant lines from “test_LVV-T1947_DiaObject.py”:
# Select an arbitrary source catalog from a diaObject table:
instrument = ‘HSC’
band = ‘i’
tract = 9697
# patch = 13
skymap = ‘hsc_rings_v1’
detector = 40
visit = 36180

# Run the test
LVVT1947(instrument, tract, detector, visit, band, skymap)

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
The script outputs the following, confirming that all “instFlux” values in the “goodSeeingDiff_diaSrc” table are in units of “counts”, and all calibrated fluxes from the “forced_src_diaObject” table and “diaSourceTable” are in units

of nJy within a reasonable range of flux values:

Output from script “test_LVV-T1947_DiaSource.py”:

lsst_distrib g4213664e8e+b08e1c1b0b w_latest w_2024_04 current setup
Input dataId: {‘instrument’: ‘HSC’, ‘visit’: 36180, ‘detector’: 40}
Checking goodSeeingDiff_diaSrc...

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 src table instFlux entries have units of counts: True

Checking forced_src_diaObject...

base_LocalBackground_instFlux..... count
base_LocalBackground_instFluxErr..... count
base_PsfFlux_instFlux..... count
base_PsfFlux_instFluxErr..... count

All src table instFlux entries have units of counts: True

Checking flux values for visit 36180

flux column % of bad flux values

apFlux 0.0 %
psfFlux 0.0 %
scienceFlux 0.0 %
dipoleMeanFlux 0.0 %

==
The second script outputs the following, confirming that all calibrated fluxes from the “diaObjectTable_tract” are in units of nJy within a reasonable range of flux values:

Output from script "test_LVV-T1947_DiaObject.py":

lsst_distrib g4213664e8e+b08e1c1b0b w_latest w_2024_04 current setup
Input dataId: {'instrument': 'HSC', 'tract': 9697, 'band': 'i', 'detector': 40, 'visit': 36180, 'skymap': 'hsc_rings_v1'}

———————————————————-

Checking flux values for tract 9697, diaObjectTable_tract

flux column % of bad flux values

g_psfFluxMAD 0.0 %
g_psfFluxMean 0.0 %
g_scienceFluxMean 0.0 %
g_psfFluxMin 0.0 %
g_psfFluxMax 0.0 %
g_psfFluxPercentile05 0.0 %
g_psfFluxPercentile25 0.0 %
g_psfFluxPercentile50 0.0 %
g_psfFluxPercentile75 0.0 %
g_psfFluxPercentile95 0.0 %
g_psfFluxSigma 0.0 %
g_scienceFluxSigma 0.0 %

flux column % of bad flux values

r_psfFluxMAD 0.0 %
r_psfFluxMean 0.0 %
r_scienceFluxMean 0.0 %
r_psfFluxMin 0.0 %
r_psfFluxMax 0.0 %
r_psfFluxPercentile05 0.0 %
r_psfFluxPercentile25 0.0 %
r_psfFluxPercentile50 0.0 %
r_psfFluxPercentile75 0.0 %
r_psfFluxPercentile95 0.0 %
r_psfFluxSigma 0.0 %

r_scienceFluxSigma 0.0 %

flux column % of bad flux values

i_psfFluxMAD 0.0 %
i_psfFluxMean 0.0 %
i_scienceFluxMean 0.0 %
i_psfFluxMin 0.0 %
i_psfFluxMax 0.0 %
i_psfFluxPercentile05 0.0 %
i_psfFluxPercentile25 0.0 %
i_psfFluxPercentile50 0.0 %
i_psfFluxPercentile75 0.0 %
i_psfFluxPercentile95 0.0 %
i_psfFluxSigma 0.0 %
i_scienceFluxSigma 0.0 %

flux column % of bad flux values

z_psfFluxMAD 0.0 %
z_psfFluxMean 0.0 %
z_scienceFluxMean 0.0 %
z_psfFluxMin 0.0 %
z_psfFluxMax 0.0 %
z_psfFluxPercentile05 0.0 %
z_psfFluxPercentile25 0.0 %
z_psfFluxPercentile50 0.0 %
z_psfFluxPercentile75 0.0 %
z_psfFluxPercentile95 0.0 %
z_psfFluxSigma 0.0 %
z_scienceFluxSigma 0.0 %

flux column % of bad flux values

y_psfFluxMAD 0.0 %
y_psfFluxMean 0.0 %
y_scienceFluxMean 0.0 %
y_psfFluxMin 0.0 %

y_psfFluxMax 0.0 %
y_psfFluxPercentile05 0.0 %
y_psfFluxPercentile25 0.0 %
y_psfFluxPercentile50 0.0 %
y_psfFluxPercentile75 0.0 %
y_psfFluxPercentile95 0.0 %
y_psfFluxSigma 0.0 %
y_scienceFluxSigma 0.0 %

The script outputs the following, confirming that all calibrated fluxes from the "forcedSourceOnDiaObject" table are in units of nJy within a reasonable range of flux values:

Output from script "test_LVV-T1947_forcedSourceOnDiaSource.py":

(lsst-scipipe-8.0.0) [jcarlin@sdfrome001 fall2023_ATC]$ python test_LVV-T1947_forcedSourceOnDiaObject.py
lsst_distrib g4213664e8e+b08e1c1b0b w_2024_04 setup
Input dataId: {'instrument': 'HSC', 'tract': 9697, 'patch': 13, 'skymap': 'hsc_rings_v1'}

———————————————————-

Checking flux values for tract 9697, forcedSourceOnDiaObjectTable

flux column % of bad flux values
psfFlux 0.0 %
psfDiffFlux 0.0 %

5.1.3.9 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:
This test case can be executed by running the scripts test_LVV-T28.py, test_LVV-T28_forcedSource.py, and test_LVV-T28_DC2.py, which are available in the test report github repository's "scripts/" directory.

The tests that confirm fluxes have "reasonable" values are checking that the fluxes, if converted to magnitudes, would result in a magnitude fainter than -5.

Detailed steps results:

Step 1.1 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 using "test_LVV-T28.py," which contains the following:

# For RC2 data on the USDF:
config = '/repo/main'
collection = 'HSC/runs/RC2/w_2023_39/DM-40985'
butler = Butler(config, collections=collection)

Step 1.2 Step Execution Status: **Pass**

Description

Identify and read an appropriate repo with processed RC2 precursor data containing coadds with the Butler.

Expected Result

Actual Result

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

# Select an arbitrary source catalog from a calexp (PVI):
instrument = 'HSC'
detector = 5
visit = 36180

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

Step 1.3 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

(lsst-scipipe-8.0.0) [jcarlin@sdfrome001 fall2023_ATC]$ python test_LVV-T28.py

lsst_distrib g4213664e8e+b08e1c1b0b w_latest w_2024_04 current setup
Input dataId: {'instrument': 'HSC', 'visit': 36180, 'detector': 5}

Checking afw src table:
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_CompensatedGaussianFlux_3_instFlux ..... count
base_CompensatedGaussianFlux_3_instFluxErr ..... count
base_CompensatedGaussianFlux_5_instFlux ..... count
base_CompensatedGaussianFlux_5_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

Checking parquet sourceTable:

Checking flux values for visit 36180

flux column	% of bad flux values
calibFlux	0.0 %
ap03Flux	0.0 %
ap06Flux	0.0 %
ap09Flux	0.0 %
ap12Flux	0.0 %
ap17Flux	0.0 %
ap25Flux	0.0 %
ap35Flux	0.0 %
ap50Flux	0.0 %
ap70Flux	0.0 %
psfFlux	0.0 %
gaussianFlux	0.0 %

All “instFlux” entries have units of counts, and all fluxes are within a reasonable range of flux values, so this test has passed.

Step 2.1 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 using "test_LVV-T28_DC2.py," which contains the following:

```
# For DC2 data on the USDF:
config = '/repo/dc2'
collection = '2.2i/runs/test-med-1/w_2023_37/DM-40870'
butler = Butler(config, collections=collection)
```

Step 2.2	Step Execution Status: **Pass**

Description
Identify and read an appropriate repo with processed DC2 precursor data containing coadds with the Butler.

Expected Result

Actual Result
By default, the test script selects the following instrument/detector/visit combination. This can be changed if desired.

```
# Select an arbitrary source catalog from a calexp (PVI):
    instrument = 'LSSTCam-imSim'
    detector = 'R30_S10'
    visit = 169765

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

Step 2.3	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
(lsst-scipipe-8.0.0) [jcarlin@sdfrome001 fall2023_ATC]$ python test_LVV-T28_DC2.py
lsst_distrib g4213664e8e+b08e1c1b0b w_latest w_2024_04 current setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 169765, 'detector': 'R30_S10'}
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_CompensatedGaussianFlux_3_instFlux ..... count
base_CompensatedGaussianFlux_3_instFluxErr ..... count
base_CompensatedGaussianFlux_5_instFlux ..... count
base_CompensatedGaussianFlux_5_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

Checking flux values for visit 169765

flux column % of bad flux values

calibFlux 0.0 %
ap03Flux 0.0 %
ap06Flux 0.0 %
ap09Flux 0.0 %
ap12Flux 0.0 %
ap17Flux 0.0 %
ap25Flux 0.0 %
ap35Flux 0.0 %
ap50Flux 0.0 %
ap70Flux 0.0 %
psfFlux 0.0 %
gaussianFlux 0.0 %

All “instFlux” entries have units of counts, and all fluxes are within a reasonable range of flux values, so this test has passed.

5.1.3.10 LVV-T142 - Verify implementation of Production Fault Tolerance

Version **1**. Status **Approved**. Open *LVV-T142* test case in Jira.

Demonstrate production systems report faults in pipeline executions and that system is able to recover. Where recovery can mean the ability to provide production artifacts for examination, return production elements ready for subsequent use, and/or reset and repeat production attempts.

Preconditions:

Execution status: **Pass**

Final comment:
Executed at the USDF using the w_2023_43 version of the Science Pipelines.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Create a HTCondor Pool at USDF

Example Code
allocateNodes.py -c 8 -m 4-00:00:00 -q roma,milano -g 900 s3df

Expected Result

Actual Result
Entered:
allocateNodes.py -c 16 -n 1 -m 4-00:00:00 -q roma,milano -g 900 s3df

The following shows the active glide-in job in the queue:
squeue -u jcarlin
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
32575038 roma glide_jc jcarlin R 12:13 1 sdfrome043

Step 2 Step Execution Status: **Pass**

Description

Submit a RC2-subset run using bps

Example Code

bps submit –output u/username/collectionname ${DRP_PIPE_DIR}/drp_pipe/pipelines/HSC/DRP-RC2_subset.yaml

Expected Result

Actual Result

We use the script "bps_step1_DM-ATC.yaml", which contains the following:

```
pipelineYaml: '${DRP_PIPE_DIR}/pipelines/HSC/DRP-RC2_subset.yaml#nightlyStep1'


project: dm_atc
campaign: LVV-T142_test


includeConfigs:
  - ${DRP_PIPE_DIR}/bps/clustering/DRP-recalibrated.yaml
  - ${DRP_PIPE_DIR}/bps/resources/HSC/DRP-RC2.yaml


payload:
  payloadName: u/jcarlin
  output: "{payloadName}/{campaign}"
  butlerConfig: /sdf/group/rubin/repo/main/butler.yaml
  inCollection: "HSC/RC2_subset/defaults"


#computeSite:
site:
  s3df:
    profile:
      condor:
        +Walltime: 7200
```

Condor backend stuff
wmsServiceClass: lsst.ctrl.bps.htcondor.HTCondorService

To submit the script, we execute the following:
bps submit bps_step1_DM-ATC.yaml

Step 3 Step Execution Status: **Pass**

Description
Cancel the job while it is in progress

Example Code
bps cancel [job id]

Expected Result

Actual Result
After the job had been executing for a few minutes, canceled it using:
bps cancel –user jcarlin
(Screen output:)
Successfully canceled job with id 'sdfrome001.sdf.slac.stanford.edu#5463544.0#1698969640'

Step 4 Step Execution Status: **Pass**

Description
Check that some output products are missing

Example Code
butler query-datasets /repo/main –collections u/username/collectionname

Expected Result

Actual Result

We know that there should be 240 calexps (6 detectors, 8 visits in each of 5 filters; 6*8*5=240) in this repository once processing has completed. Use a butler query to see how many exist:

butler query-datasets /repo/main calexp –collections u/jcarlin/LVV-T142_test/20231102T233609Z | grep 'calexp' | wc
    13   104   1716

The first entry says that there are only 13 lines returned by that query. Thus some output products must be missing.

Step 5	Step Execution Status: **Pass**

Description

Submit a new job to finish the tasks that were not completed

Example Code

bps submit –output u/username/collectionname ${DRP_PIPE_DIR} drp_pipe/pipelines/HSC/DRP-RC2_subset.yaml

Expected Result

Actual Result

Resubmit the job, giving the output collection from the previous run as an input to this run:

allocateNodes.py -c 16 -n 1 -m 4-00:00:00 -q roma,milano -g 900 s3df
bps restart –id /sdf/home/j/jcarlin/u/SVV/fall2023_ATC/submit/u/jcarlin/LVV-T142_test/20231102T233609Z

Job executed, and we waited until it successfully completed.

Step 6	Step Execution Status: **Pass**

Description

Confirm that all expected files are now present

Example Code

butler query-datasets /repo/main –collections u/username/collectionname

Expected Result

Actual Result

Run the same query as before, but specifying the exact collection that the results were saved into:

butler query-datasets /repo/main calexp –collections u/jcarlin/LVV-T142_test/20231102T233609Z | grep 'calexp' | wc

240 1920 31680

We see that this execution brought the total to 240 calexps, as expected. We have thus demonstrated that the batch processing system (BPS) at the USDF successfully recovers jobs that fail before they have completed.

5.1.3.11 LVV-T1748 - Verify calculation of median error in absolute position for RA, Dec axes

Version **1**. Status **Approved**. Open *LVV-T1748* test case in Jira.

Verify that the DM system has provided the code to calculate the median error in absolute position for each axis, RA and DEC, and assess whether it meets the requirement that it shall be less than **AA1 = 50 milliarcseconds**.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Identify a dataset containing processed data.

Expected Result

A dataset that has been ingested into a Butler repository.

Actual Result

For this test we use the regularly-reprocessed HSC RC2; in particular, the data reprocessed with weekly pipelines version 'w_2024_02', in Butler collection "HSC/runs/RC2/w_2024_02/DM-42454" at the USDF.

Step 2 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

$ eups list -s lsst_distrib
  g4213664e8e+b08e1c1b0b   w_latest w_2024_04 current setup

Step 3 Step Execution Status: **Pass**

Description

Execute 'analysis_tools' 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 $ANALYSIS_TOOLS_DIR/pipelines/matchedVisitQualit
-d "band in ('g', 'r', 'i') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC'" –output u/username/a-
tools_metrics -i HSC/runs/RC2/w_2023_36 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w36_2023_tract9813_atools.tx

---

Expected Result

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

Actual Result

First we ran the "coaddQualityCore" pipeline from 'analysis_tools' on all three available tracts:

pipetask run -b /repo/main -i HSC/runs/RC2/w_2024_02/DM-42454 -p ${ANALYSIS_TOOLS_DIR}/pipelines/coaddQualityCore.yaml
-o u/jcarlin/test_LVV-T1748 –instrument lsst.obs.subaru.HyperSuprimeCam –register-dataset-types -d "tract in (9813,
9615, 9697) AND skymap='hsc_rings_v1'" 2>&1 | tee atools_LVV-T1748_test.log

Next we ran the "visitQualityCore" pipeline on a single RC2 visit:

pipetask run -b /repo/main -i HSC/runs/RC2/w_2024_02/DM-42454 -p ${ANALYSIS_TOOLS_DIR}/pipelines/visitQualityCore.yaml
-o u/jcarlin/test_LVV-T1748_visit –instrument lsst.obs.subaru.HyperSuprimeCam –register-dataset-types -d "visit
in (36180) AND skymap='hsc_rings_v1' AND instrument='HSC'" 2>&1 | tee atools_LVV-T1748_test_visit.log

Both of these pipelines contain tasks that will calculate metrics based on matching to the Gaia DR3 reference catalog.

Step 4 Step Execution Status: **Pass**

Description

Confirm that the metric AA1 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 AA1 has been calculated.

Actual Result

Open a python prompt, then instantiate the Butler and read the metric bundles that were created by the above runs of 'analysis_tools':

```
from lsst.daf.butler import Butler
repo = '/repo/main'
collection = 'u/jcarlin/test_LVV-T1748'
butler = Butler(repo, collections=collection)
did_tract = {'instrument':'HSC', 'tract':9615, 'skymap':'hsc_rings_v1'}
metric_extract = butler.get('objectTable_tract_gaia_dr3_20230707_match_metrics', collections=collection, dataId=did_tract)

for met in metric_extract['astromDiffMetrics']:
    print(met)
```

```
g_AA1_RA_coadd: 2.1063837266410697 mas
g_AA1_sigmaMad_RA_coadd: 3.9587443763583363 mas
g_AA1_Dec_coadd: 2.1491007237806055 mas
g_AA1_sigmaMad_Dec_coadd: 4.334318369967101 mas
g_AA1_tot_coadd: 5.557193272612054 mas
g_AA1_sigmaMad_tot_coadd: 3.8598538443562407 mas
r_AA1_RA_coadd: 2.7165050880739723 mas
r_AA1_sigmaMad_RA_coadd: 4.520586827619122 mas
r_AA1_Dec_coadd: 2.4681338712806418 mas
r_AA1_sigmaMad_Dec_coadd: 4.641432559314666 mas
r_AA1_tot_coadd: 6.305816679054206 mas
r_AA1_sigmaMad_tot_coadd: 4.120642210353908 mas
i_AA1_RA_coadd: 3.519391958661799 mas
i_AA1_sigmaMad_RA_coadd: 4.326955084103768 mas
```

i_AA1_Dec_coadd: 3.3291494215381685 mas
i_AA1_sigmaMad_Dec_coadd: 4.404755426441496 mas
i_AA1_tot_coadd: 6.952576143474123 mas
i_AA1_sigmaMad_tot_coadd: 4.036796559569423 mas
z_AA1_RA_coadd: 2.41553204318734 mas
z_AA1_sigmaMad_RA_coadd: 4.428035048189116 mas
z_AA1_Dec_coadd: 2.1116089754497076 mas
z_AA1_sigmaMad_Dec_coadd: 4.641887604460565 mas
z_AA1_tot_coadd: 6.039798859474133 mas
z_AA1_sigmaMad_tot_coadd: 4.182977418904133 mas
y_AA1_RA_coadd: 2.408819972288256 mas
y_AA1_sigmaMad_RA_coadd: 4.440931123927884 mas
y_AA1_Dec_coadd: 2.1116089754497076 mas
y_AA1_sigmaMad_Dec_coadd: 4.626056674759659 mas
y_AA1_tot_coadd: 6.0289912626929 mas
y_AA1_sigmaMad_tot_coadd: 4.1957238417228 mas

```
did_visit = {'instrument':'HSC', 'visit':36180, 'skymap':'hsc_rings_v1'}
collection_vis = 'u/jcarlin/test_LVV-T1748_visit'
butler_vis = Butler(repo, collections=collection_vis)
metric_extract_vis = butler.get('sourceTable_visit_gaia_dr3_20230707_match_metrics', collections=collection_vis, dataId=did_visit)
for met in metric_extract_vis['astromDiffMetrics']:
    print(met)
```

AA1_RA: 0.2177698661398608 mas
AA1_sigmaMad_RA: 2.6860945686629476 mas
AA1_Dec: 0.1395363322886922 mas
AA1_sigmaMad_Dec: 2.801517008917446 mas
AA1_tot: 3.243461716402636 mas
AA1_sigmaMad_tot: 1.9738141749210063 mas

We see that the calculated values of AA1 are well below the required 50 mas threshold in both the coadds and the single-visit metrics. We thus deem this test to have passed.

5.1.3.12 LVV-T1759 - Verify that the repeatability outlier limit for isolated bright non-saturated point sources in the g, r, and i filters (PA2gri) can be applied.

Version **1**. Status **Approved**. Open *LVV-T1759* test case in Jira.

Verify that the DM system has provided the code to apply the repeatability outlier limit for isolated bright non-saturated point sources in the g, r, and i filters(PA2gri) to to computed values of the PF1 metric.

Preconditions:

Execution status: **Pass**

Final comment:
This test used a modified version of the analysis_tools pipeline "matchedVisitQualityCore.yaml."

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify a dataset containing at least one field in each of the g, r, and i filters with multiple overlapping visits.

Expected Result
A dataset that has been ingested into a Butler repository.

Actual Result
For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_35 pipelines.

Step 2 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

Current weekly stack initialized:
lsst_distrib g4213664e8e+7835acb1bb w_latest current w_2023_40 setup

Step 3	Step Execution Status: **Pass**

Description

Execute 'analysis_tools' 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 $ANALYSIS_TOOLS_DIR/pipelines/matchedVisitQualit
-d "band in ('g', 'r', 'i') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC'" –output u/username/a-
tools_metrics -i HSC/runs/RC2/w_2023_36 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w36_2023_tract9813_atools.tx

Expected Result

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

Actual Result

Changed the value of PA2 in matchedVisitQualityCore_changePA2.yaml to 15.0:
atools.stellarPhotometricRepeatability.PA2Value: 15.0

Then, executed the pipeline as follows:
pipetask –long-log run -j 2 -b /repo/main –register-dataset-types -p matchedVisitQualityCore_changePA2.yaml -d “band in (’g’, ‘r’, ‘i’) AND tract=9813 AND skymap=’hsc_rings_v1’ AND instrument=’HSC’” –output u/jcarlin/pa2_15 -i HSC/runs/RC2/w_2023_35/DM-40588
–instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w35_2023_tract9813_pa2_15.txt

---

Step 4 Step Execution Status: **Pass**

Description

Confirm that the PA2gri threshold has been applied to the assessment of the computed values of PF1 for filters g,r,i.

Expected Result

A JSON file (and/or a report generated from that JSON file) demonstrating that PF1 has been calculated (and that it used the requested threshold value of PA2gri).

Actual Result

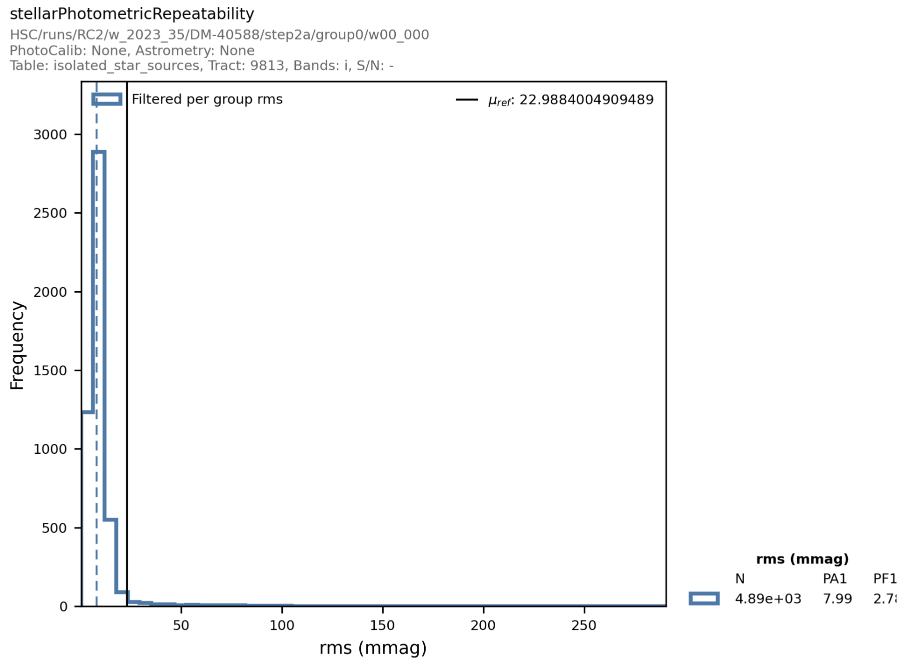
Examine the results in a python terminal:

```
>>> from lsst.daf.butler import Butler
>>> repo='/repo/main'
>>> collection='u/jcarlin/pa2_15'
>>> butler = Butler(repo, collections=collection)
>>> dataId = {'tract':9813, 'instrument':'HSC', 'skymap':'hsc_rings_v1'}
>>> metrics = butler.get('matchedVisitCore_metrics', dataId=dataId)
>>> for m in metrics['stellarPhotometricRepeatability']:
... print(m)
...
g_stellarPhotRepeatStdev: 6.979476571776661 mmag
g_stellarPhotRepeatOutlierFraction: 3.890690134321445 %
g_ct: 2159.0 ct
r_stellarPhotRepeatStdev: 8.902557135179485 mmag
r_stellarPhotRepeatOutlierFraction: 4.64067886502254 %
r_ct: 3771.0 ct
i_stellarPhotRepeatStdev: 7.9884004909489015 mmag
i_stellarPhotRepeatOutlierFraction: 2.779480891068874 %
i_ct: 4893.0 ct


>>> metrics_config = butler.get('analyzeMatchedVisitCore_config', dataId=dataId)
>>> config_dict = metrics_config.toDict()
```

>>> config_dict['atools']['stellarPhotometricRepeatability']['process']['calculateActions']['photRepeatOutlier']
{'op': 'ge', 'threshold': 15.0, 'vectorKey': 'perGroupStdevFiltered', 'percent': True, 'relative_to_median': True}

The following figure is output by the pipeline. The threshold (vertical black line) is set at 15 mmag above the measured PA1 value, as expected.



Step 5 Step Execution Status: **Pass**

Description

Change the value of the PA2 threshold in the pipeline yaml for analysis_tools, then rerun analysis_tools

Expected Result

Actual Result

Changed the following line in matchedVisitQualityCore_changePA2.yaml:
atools.stellarPhotometricRepeatability.PA2Value: 25.0

then executed:
pipetask –long-log run -j 2 -b /repo/main –register-dataset-types -p matchedVisitQualityCore_changePA2.yaml -d "band in ('g', 'r', 'i') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC'" –output u/jcarlin/pa2_25 -i HSC/runs/RC2/w_2023_35/DM-40588 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w35_2023_tract9813_pa2_25.txt

Step 6 Step Execution Status: **Pass**

Description

Confirm that the new PA2 threshold has been applied when computing PF1.

Expected Result

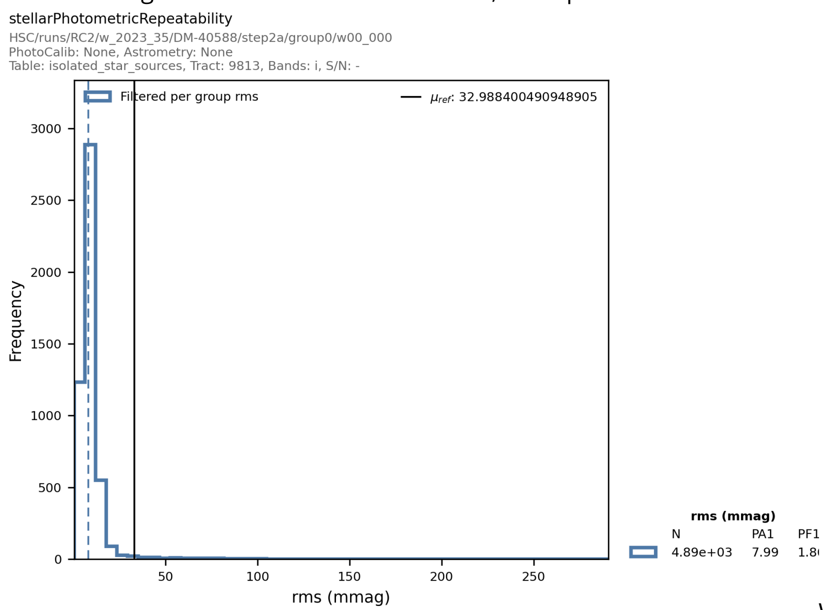
A JSON file (and/or a report generated from that JSON file) demonstrating that PF1 has been calculated (and that it used the requested threshold value of PA2gri).

Actual Result

Open a python terminal to check the results:

```
>>> from lsst.daf.butler import Butler
>>> repo='/repo/main'
>>> collection='u/jcarlin/pa2_25'
>>> butler = Butler(repo, collections=collection)
>>> dataId = {'tract':9813, 'instrument':'HSC', 'skymap':'hsc_rings_v1'}
>>> metrics = butler.get('matchedVisitCore_metrics', dataId=dataId)
>>> for m in metrics['stellarPhotometricRepeatability']:
...     print(m)
...
g_stellarPhotRepeatStdev: 6.979476571776661 mmag
g_stellarPhotRepeatOutlierFraction: 3.1032885595182953 %
g_ct: 2159.0 ct
r_stellarPhotRepeatStdev: 8.902557135179485 mmag
r_stellarPhotRepeatOutlierFraction: 3.07610713338637 %
r_ct: 3771.0 ct
i_stellarPhotRepeatStdev: 7.9884004909489015 mmag
i_stellarPhotRepeatOutlierFraction: 1.859799713876967 %
i_ct: 4893.0 ct
z_stellarPhotRepeatStdev: nan mmag
z_stellarPhotRepeatOutlierFraction: nan %
z_ct: 0.0 ct
y_stellarPhotRepeatStdev: nan mmag
y_stellarPhotRepeatOutlierFraction: nan %
y_ct: 0.0 ct
>>> metrics_config = butler.get('analyzeMatchedVisitCore_config', dataId=dataId)
>>> config_dict = metrics_config.toDict()
>>> config_dict['atools']['stellarPhotometricRepeatability']['process']['calculateActions']['photRepeatOutlier']
{'op': 'ge', 'threshold': 25.0, 'vectorKey': 'perGroupStdevFiltered', 'percent': True, 'relative_to_median': True}
```

We can see from both the butler artifacts seen above, and the figure below, that the threshold PA2 has been set to 25.0 mmag from the measured PA1 value, as requested.



We have thus demonstrated that the repeatability outlier limit PA2 can be applied for the gri bands.

5.1.3.13 LVV-T1758 - Verify that the repeatability outlier limit for isolated bright non-saturated point sources in the u, z, and y filters (PA2uzy) can be applied.

Version **1**. Status **Approved**. Open *LVV-T1758* test case in Jira.

Verify that the DM system has provided the code to apply the repeatability outlier limit for isolated bright non-saturated point sources in the u, z, and y filters(PA2uzy) to computed values of the PF1 metric.

Preconditions:

Execution status: **Pass**

Final comment:
Note that because we do not have access to u-band data, this test was performed for only y- and z-band. The steps would be unchanged for u-band data.

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

For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_35 pipelines.

Step 2 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
Current weekly stack initialized:
lsst_distrib g4213664e8e+7835acb1bb w_latest current w_2023_40 setup

---

Step 3 Step Execution Status: **Pass**

Description
Execute 'analysis_tools' 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 $ANALYSIS_TOOLS_DIR/pipelines/matchedVisitQualit
-d "band in ('g', 'r', 'i') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC"' –output u/username/a-
tools_metrics -i HSC/runs/RC2/w_2023_36 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w36_2023_tract9813_atools.tx

---

Expected Result
The output collection (in this case, "u/username/atools_metrics") containing metric measurements and any associated extras and metadata is available via the butler.

Actual Result
Changed the value of PA2 in matchedVisitQualityCore_changePA2.yaml to 15.0:
atools.stellarPhotometricRepeatability.PA2Value: 15.0

Then, executed the pipeline as follows:
pipetask –long-log run -j 2 -b /repo/main –register-dataset-types -p matchedVisitQualityCore_changePA2.yaml -d
"band in ('z', 'y') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC"' –output u/jcarlin/pa2_15_zy -i
HSC/runs/RC2/w_2023_35/DM-40588 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w35_2023_tract9813_zy_pa2_15.tx

---

Step 4 Step Execution Status: **Pass**

Description
Confirm that the PA2uzy threshold has been applied to the assessment of the computed values of PF1 for filters

u,z,y.

Expected Result

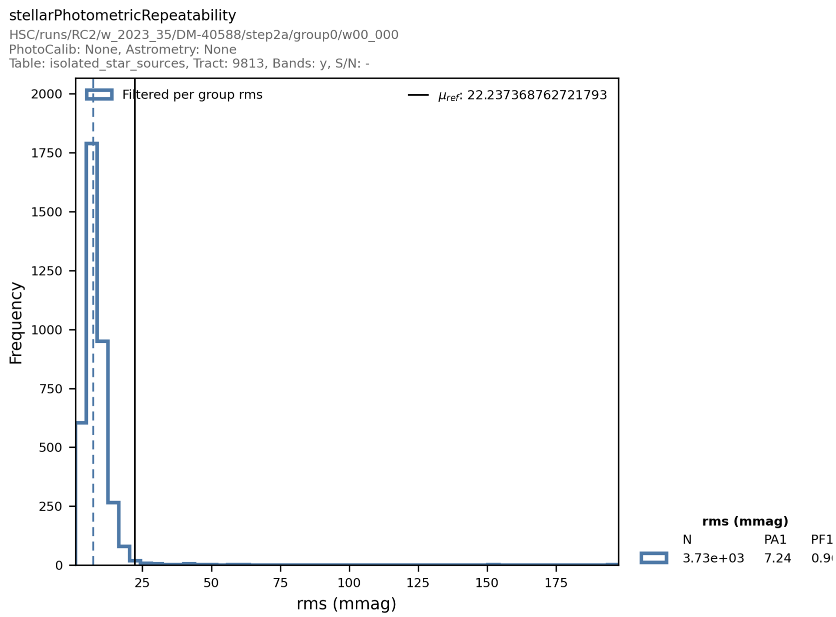
A JSON file (and/or a report generated from that JSON file) demonstrating that PF1 has been calculated (and that it used the requested PA2uzy threshold).

Actual Result

Examine the results in a python terminal:

```
>>> from lsst.daf.butler import Butler
>>> repo='/repo/main'
>>> collection='u/jcarlin/pa2_15_zy'
>>> butler = Butler(repo, collections=collection)
>>> dataId = {'tract':9813, 'instrument':'HSC', 'skymap':'hsc_rings_v1'}
>>> metrics = butler.get('matchedVisitCore_metrics', dataId=dataId)
>>> for m in metrics['stellarPhotometricRepeatability']:
... print(m)
...
z_stellarPhotRepeatStdev: 6.924991284071611 mmag
z_stellarPhotRepeatOutlierFraction: 1.9053231192300137 %
z_ct: 5091.0 ct
y_stellarPhotRepeatStdev: 7.237368762721793 mmag
y_stellarPhotRepeatOutlierFraction: 0.964371818912403 %
y_ct: 3733.0 ct
>>> metrics_config = butler.get('analyzeMatchedVisitCore_config', dataId=dataId)
>>> config_dict = metrics_config.toDict()
>>> config_dict['atools']['stellarPhotometricRepeatability']['process']['calculateActions']['photRepeatOutlier']
{'op': 'ge', 'threshold': 15.0, 'vectorKey': 'perGroupStdevFiltered', 'percent': True, 'relative_to_median': True}
```

The following figure is output by the pipeline. The threshold (vertical black line) is set at 15 mmag above the measured PA1 value, as expected.



Step 5	Step Execution Status: **Pass**

Description

Change the value of the PA2 threshold in the pipeline yaml for analysis_tools, then rerun analysis_tools

Expected Result

Actual Result

Changed the following line in matchedVisitQualityCore_changePA2.yaml:
atools.stellarPhotometricRepeatability.PA2Value: 25.0

then executed:
pipetask –long-log run -j 2 -b /repo/main –register-dataset-types -p matchedVisitQualityCore_changePA2.yaml -d "band in ('z', 'y') AND tract=9813 AND skymap='hsc_rings_v1' AND instrument='HSC"' –output u/jcarlin/pa2_25_zy -i HSC/runs/RC2/w_2023_35/DM-40588 –instrument lsst.obs.subaru.HyperSuprimeCam 2>&1 | tee w35_2023_tract9813_zy_pa2_25.tx

Step 6	Step Execution Status: **Pass**

Description

Confirm that the new PA2 threshold has been applied when computing PF1.

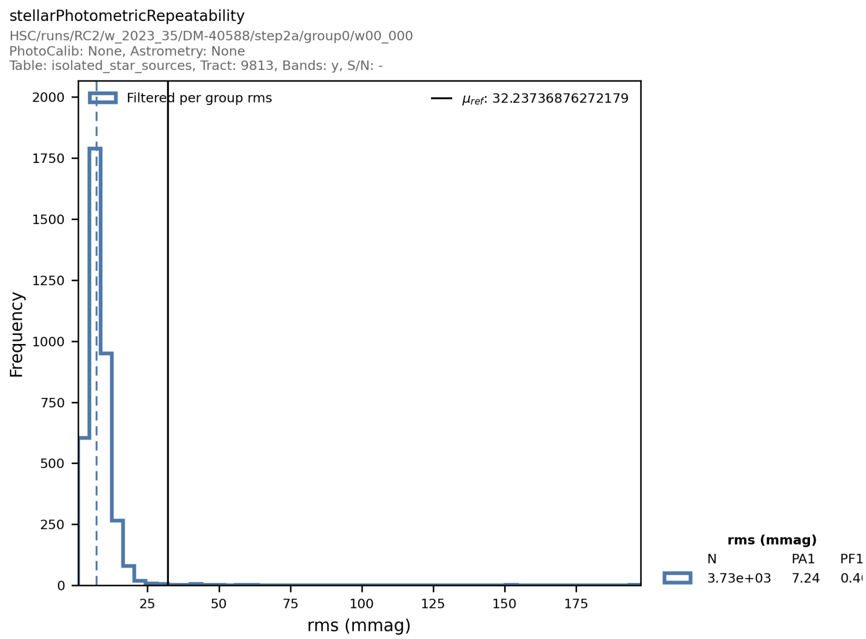
Expected Result

A JSON file (and/or a report generated from that JSON file) demonstrating that PF1 has been calculated (and that it used the requested threshold value of PA2gri).

Actual Result

```
>>> from lsst.daf.butler import Butler
>>> repo='/repo/main'
>>> collection='u/jcarlin/pa2_15_zy'
>>> butler = Butler(repo, collections=collection)
>>> dataId = {'tract':9813, 'instrument':'HSC', 'skymap':'hsc_rings_v1'}
>>> metrics = butler.get('matchedVisitCore_metrics', dataId=dataId)
>>> for m in metrics['stellarPhotometricRepeatability']:
... print(m)
...
z_stellarPhotRepeatStdev: 6.924991284071611 mmag
z_stellarPhotRepeatOutlierFraction: 1.9053231192300137 %
z_ct: 5091.0 ct
y_stellarPhotRepeatStdev: 7.237368762721793 mmag
y_stellarPhotRepeatOutlierFraction: 0.964371818912403 %
y_ct: 3733.0 ct
>>> metrics_config = butler.get('analyzeMatchedVisitCore_config', dataId=dataId)
>>> config_dict = metrics_config.toDict()
>>> config_dict['atools']['stellarPhotometricRepeatability']['process']['calculateActions']['photRepeatOutlier']
{'op': 'ge', 'threshold': 25.0, 'vectorKey': 'perGroupStdevFiltered', 'percent': True, 'relative_to_median': True}
```

We can see from both the butler artifacts seen above, and the figure below, that the threshold PA2 has been set to 25.0 mmag from the measured PA1 value, as requested.



We have thus demonstrated that the repeatability outlier limit PA2 can be applied for the z and y bands.

5.1.3.14 LVV-T149 - Verify implementation of Catalog Queries

Version **1**. Status **Approved**. Open *LVV-T149* test case in Jira.

Verify that SQL, or a similar structured language, can be used to query catalogs.

Preconditions:
An operational QSERV database that has been verified via LVV-T1085 and LVV-T1086 and LVV-T1087.

Execution status: **Pass**

Final comment:
Executed using the IDF Notebook, Portal, and API aspects. For the notebook execution, we used science pipelines version w_2023_34.

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute a simple query (for example, the one below) and confirm that it returns the expected result.

Example Code

SELECT * FROM dp02_dc2_catalogs.Object as obj WHERE CONTAINS(POINT('ICRS', obj.coord_ra, obj.coord_dec), CIRCLE('ICRS', 62.0, -37.0, 0.10)) = 1

Expected Result

A catalog of objects satisfying the specified constraints. The catalog should contain 26,115 results.

Actual Result

The query was executed in the notebook "test_LVV-T149.ipynb" attached to this document's Github repository. The query executed successfully in the notebook, and returned 26,115 results, as expected.

Step 2	Step Execution Status: **Pass**

Description

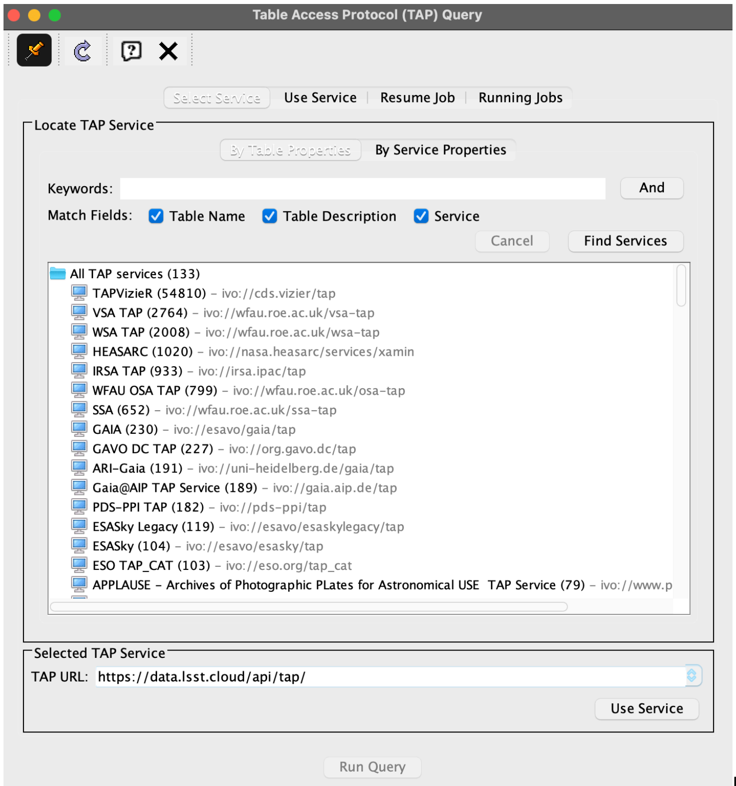
Repeat the query from all available access routes (e.g., an external VO client, the Science Platform query tool, and from within the Notebook Aspect), confirming in each case that the results are as expected.

Expected Result

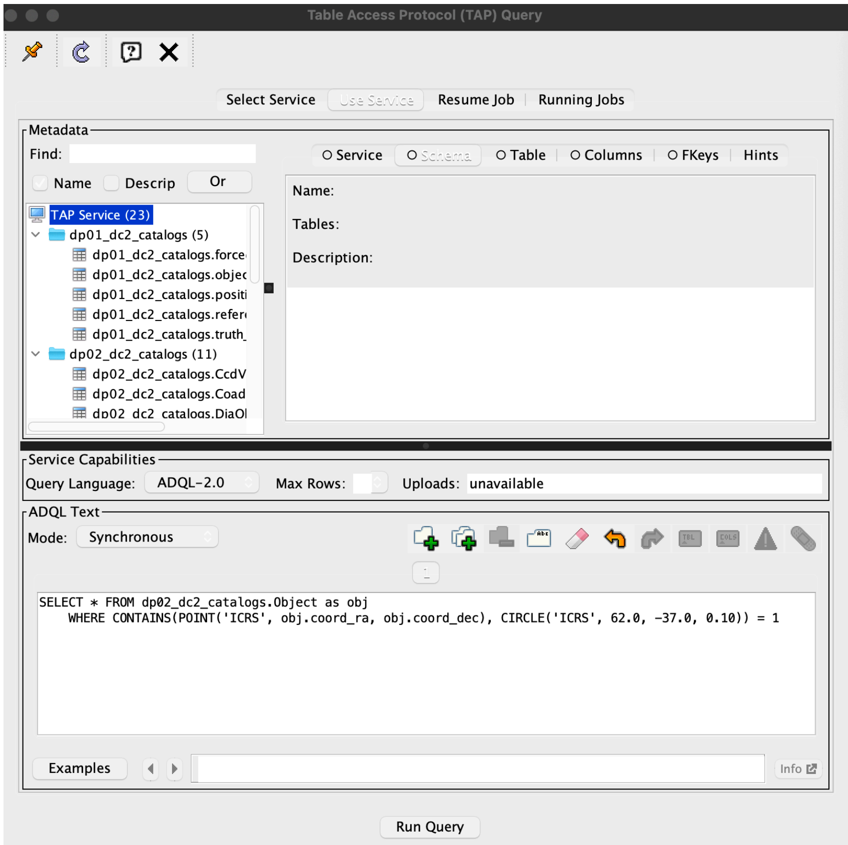
Actual Result

The Notebook query was demonstrated in step 1.

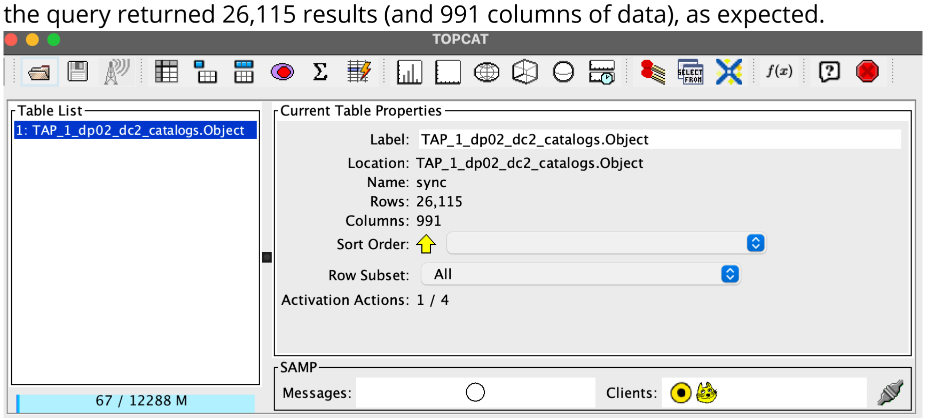
To query via the API aspect, we use Topcat, entering "https://data.lsst.cloud/api/tap/" in the "Select TAP service" window as seen below. (Note that the user has already set up an access token for Topcat to access the RSP.)



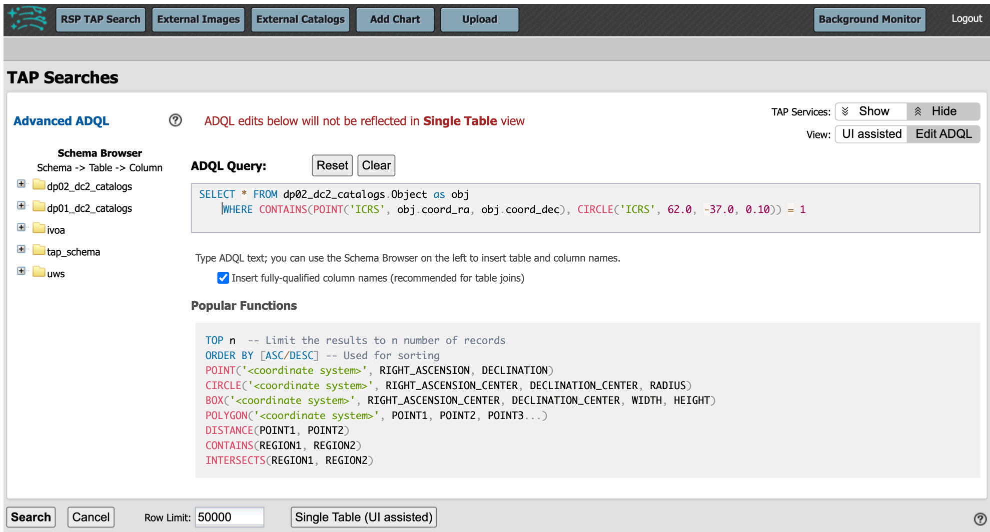
Next we execute the query from the "Use Service" window as follows:



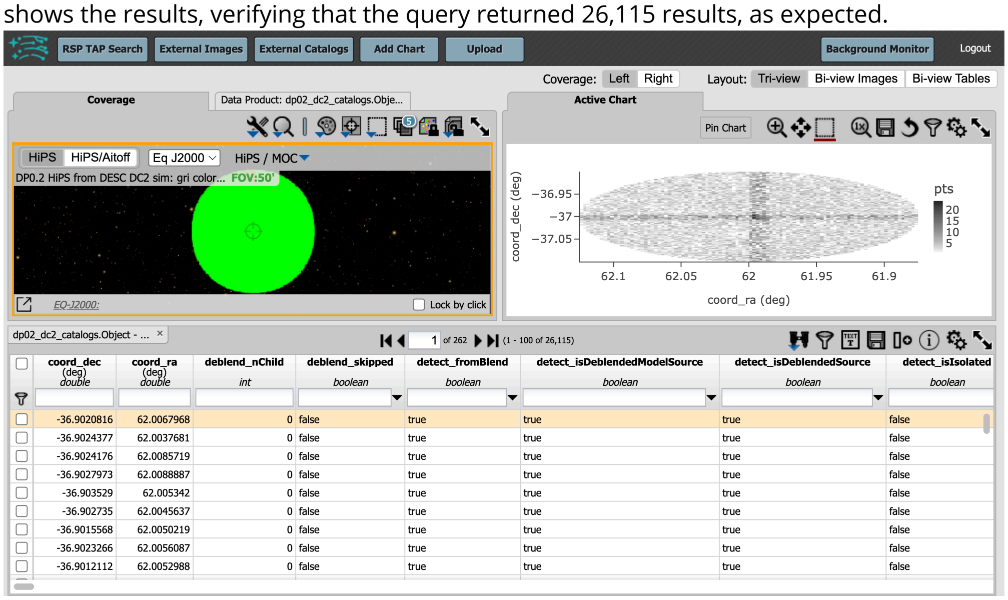
The following screenshot shows that the query returned 26,115 results (and 991 columns of data), as expected.



Next we execute the same query from the RSP Portal aspect.



The following screenshot shows the results, verifying that the query returned 26,115 results, as expected.



We have thus demonstrated that catalogs can be queried via the Notebook, Portal, and API aspects using ADQL queries.

5.1.3.15 LVV-T40 - Verify implementation of Generate WCS for Visit Images

Version **1**. Status **Approved**. Open *LVV-T40* test case in Jira.

Verify that Processed Visit Images produced by the AP and DRP pipelines include FITS WCS accurate to specified **astrometricAccuracy** over the bounds of the image.

Preconditions:

Execution status: **Pass**

Final comment:
Test executed with science pipelines version w_2023_37 in the RSP Notebook aspect at the USDF.

The executed notebook was saved in the repository associated with this campaign's test report as "notebooks/test_LVV-T40_T1240.ipynb".

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Identify an appropriate repo containing processed HSC data for this test.

Expected Result
A dataset with Processed Visit Images available.

Actual Result
For this test we use the most recent reprocessing of the Subaru+HSC RC2 dataset. The data were processed with the w_2023_32 pipelines.

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 access is as follows:

repo = '/repo/main'

rc2_collection = 'HSC/runs/RC2/w_2023_32/DM-40356'

butler = Butler(repo, collections=rc2_collection)

Step 3 Step Execution Status: **Pass**

Description

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

Expected Result

A table containing detected sources, and a WCS object associated with that catalog.

Actual Result

This was done 500 times, extracting the source list and WCS for 500 randomly-selected CCD/visit combinations from the repository.

Step 4 Step Execution Status: **Pass**

Description

Confirm that each CCD within the visit image contains at least **astrometricMinStandards** astrometric standards that were used in deriving the astrometric solution.

Expected Result

At least **astrometricMinStandards** from each CCD were used in determining the WCS solution.

Actual Result

It was confirmed that all CCDs selected had more than astrometricMinStandards=5 standards used in their WCS solutions.

This was done using the following code to extract the number of astrometric standards for each image:

```
astrom_selection = np.where(src['calib_astrometry_used'] == True)
num_calib_astrom.append(np.size(astrom_selection))
```

In the end, we calculate the fraction of fields that met this requirement, using:
wcsFlagsPercent = (np.size(np.where(num_calib_astrom > 5))/np.size(num_calib_astrom))*100.0*u.percent

The result (from the notebook) is:
Percentage of fields with > astrometricMinStandards=5: 100.0 % – True

---

Step 5 Step Execution Status: **Pass**

Description
Starting from the XY pixel coordinates of the sources, apply the WCS to obtain RA, Dec coordinates.

Expected Result
A list of RA, Dec coordinates for all sources in the catalog.

Actual Result
Executed the following (for each CCD/visit) to create a list of RA, Dec coords from XY:

```
xxx = src.getX()
yyy = src.getY()
radec = [wcs.pixelToSky(xxx[i], yyy[i]) for i in range(len(xxx))]
radec_arr = np.array([(coo.getRa().asDegrees(), coo.getDec().asDegrees()) for coo in radec])
```

This yields an array with RA, Dec coordinates.

Step 6 Step Execution Status: **Pass**

Description
We will assume that Gaia provides a source of "truth." Match the source list to Gaia DR3, and calculate the positional offset between the test data and the Gaia catalog.

Expected Result

A matched catalog of sources in common between the test source list and Gaia DR3.

Actual Result

Used astroquery to extract Gaia sources, then Astropy utilities to match the catalogs:

```
gaia_mch = Gaia.query_object_async(coordinate=cen, width=width, height=height)
sc_src = SkyCoord(radec_arr[:,0]*u.deg, radec_arr[:,1]*u.deg)
sc_gaia = SkyCoord(gaia_mch['ra'], gaia_mch['dec'])
src_match = sc_src.match_to_catalog_sky(sc_gaia)
sep_match = src_match[1]
```

Filtered the matched catalog to keep only matches with <0.5" separation and with magnitude difference < 1.0 (relative to the median magnitude difference of all sources, to account for different filters):

```
okmch = (sep_match.arcsec < 0.5)
matchsep = sep_match[okmch]
# Require the matches to have similar magnitudes:
gaia_gmag = gaia_mch['phot_g_mean_mag']
magdiff = src_mag[okmch][:,0]-gaia_gmag[src_match[0][okmch]]
okmagdiff = (np.abs(magdiff - np.median(magdiff)) < 1.0)
okmatchsep = matchsep[okmagdiff]
```

This yields the final matched list.

Step 7	Step Execution Status: **Pass**

Description

Apply appropriate cuts to extract the optimal dataset for comparison, then calculate statistics (median, 1-sigma range, etc.; also plot a histogram) of the offsets in milliarcseconds. Confirm that the offset is less than **astrometricAccuracy**.

Expected Result

Histogram and relevant statistics needed to confirm that the WCS transformation is accurate.

Actual Result

Figures shown in the notebook. Rather than histograms, we used comparisons of the various extracted parameters.

In addition to figures, we calculated the percentage of images that satisfied the requirement on astrometricAccuracy. This was less than 100% in all trials, likely due to some deep (or problematic) images having few Gaia matches. The results printed in the notebook are as follows:

```
Percentage of fields meeting the threshold:  95.99198396793587 %  --  False
```

Some brief exploration into reasons for a few images having large astrometric residuals is included in the notebook ('test_LVV-T40_T1240.ipynb'). These explorations suggest that the small fraction of "failing" images are not meeting the requirement due to inherent limitations in the data, and not a problem with the DM algorithms. Thus, we grant this test a "Pass."

Step 8	Step Execution Status: **Pass**

Description

Repeat Step 5, but for subregions of the image, to confirm that the accuracy criterion is met at all positions.

Expected Result

astrometricAccuracy requirement is met over the entire image.

Actual Result

Upon examination, we find that many images have only <50 Gaia matches over the entire frame. This is too few stars to get statistically meaningful results from subregions, so we did not perform this portion of the test.

We denote this test step as "Pass," as it is likely the requirement text will need to be removed or changed to account for the paucity of Gaia sources in many fields, and it is thus likely that this step will not be executed in the future. Nonetheless, this particular step does not have any bearing on the overall requirement being tested.

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 (IVOA standard)
AP	Alert Production
API	Application Programming Interface
BPS	Batch Production Service
CCD	Charge-Coupled Device
CI	Continuous Integration
DC2	Data Challenge 2 (DESC)
DEC	Declination
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
DR3	Data Release 3
DRP	Data Release Production
FITS	Flexible Image Transport System
FTS3	File Transfer Service 3
HSC	Hyper Suprime-Cam
ICRS	International Celestial Reference Frame
IDF	Interim Data Facility
JSON	JavaScript Object Notation
LDM	LSST Data Management (Document Handle)
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
NFS	Network File System
OCS	Observatory Control System
ObsLocTAP	Observation Locator Table Access Protocol (IVOA standard)
PMCS	Project Management Controls System
PSF	Point Spread Function
PVI	Processed Visit Image
PanDA	Production ANd Distributed Analysis system
QC	Quality Control
QSERV	LSST Query Services
RA	Right Ascension
RSP	Rubin Science Platform
S3	(Amazon) Simple Storage Service
SQL	Structured Query Language
TAP	Table Access Protocol (IVOA standard)
US	United States
USDF	United States Data Facility
VO	Virtual Observatory
WCS	World Coordinate System
bps	bit(s) per second