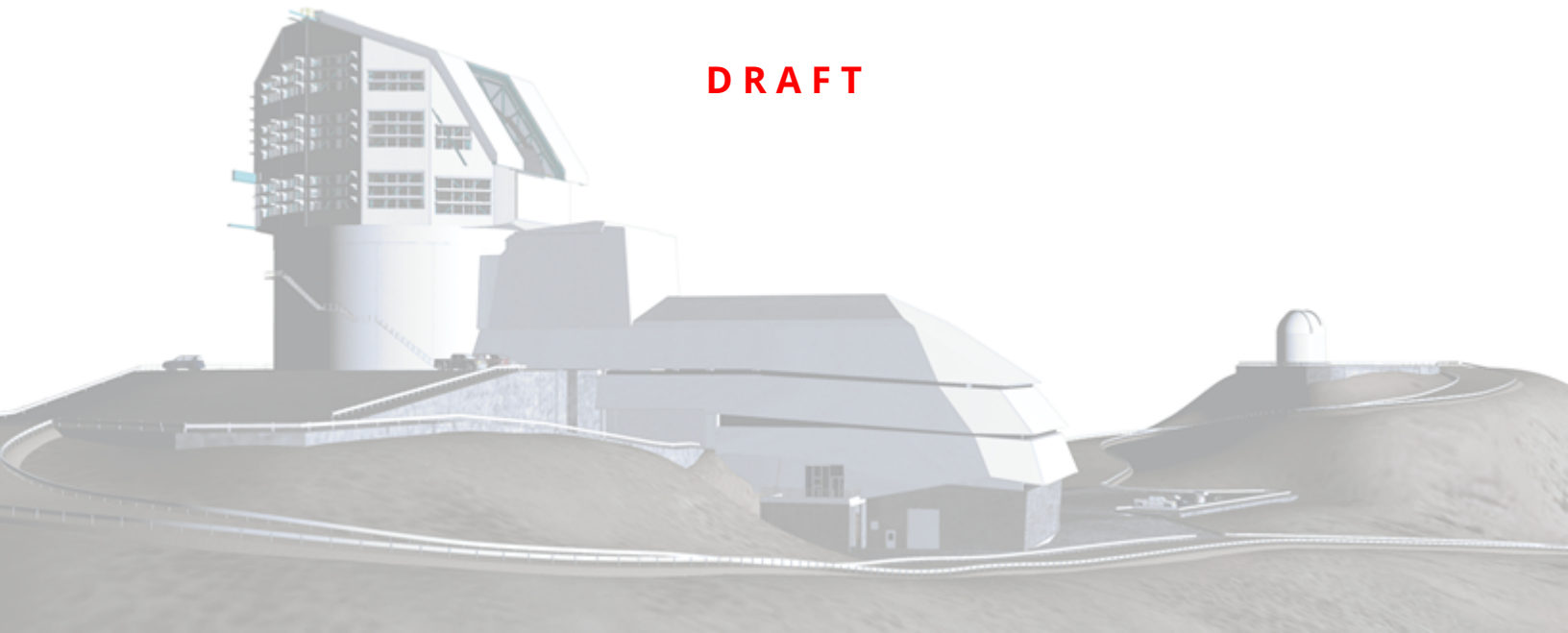

Vera C. Rubin Observatory
Data Management

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

Jeffrey Carlin

DMTR-371

Latest Revision: 2022-08-17

DRAFT

Abstract

This is the test plan and report for **Data Management Acceptance Test Campaign 1**, an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2022-08-17 . The most recent change to the document repository was on 2022-08-23.

Change Record

Version	Date	Description	Owner name
	2022-07-12	First draft	Jeff Carlin

Document curator: Jeff Carlin
Document source location: `https://github.com/lsst-dm/DMTR-371`
Version from source repository: 9b7de80

Contents

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

1 Introduction

1.1 Objectives

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

1.2 System Overview

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

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

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

1.3 Document Overview

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

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

Section 4 provides a summary of the test results, including an overview in Table 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-P99 in Jira is **Draft** .

1.4 References

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

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

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

[4] **[LDM-554]**, Dubois-Felsmann, G., Ciardi, D., Mueller, F., Economou, F., 2019, *Data Management LSST Science Platform Requirements*, LDM-554, URL https://ldm-554.lsst.io/, Vera C. Rubin Observatory Data Management Controlled Document

[5] **[LDM-639]**, Guy, L., Wood-Vasey, W., Bellm, E., et al., 2020, *LSST Data Management Acceptance Test Specification*, LDM-639, URL https://ldm-639.lsst.io/, Vera C. Rubin Observatory Data Management Controlled Document

[6] **[LDM-503]**, O'Mullane, W., Swinbank, J., Juric, M., et al., 2021, *Data Management Test Plan*,

LDM-503, URL https://ldm-503.lsst.io/,
Vera C. Rubin Observatory Data Management Controlled Document

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

2 Test Plan Details

2.1 Data Collection

Observing is not required for this test campaign.

2.2 Verification Environment

Most testing will be performed using the Rubin Science Platform (RSP), hosted at the IDF, with auxTel-related tests using the development cluster at the USDF. In particular, we will use version X_XXXX_XX 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-P99 owner:	**Jeffrey Carlin**	
	T. Cycle LVV-C208 owner:	**Jeffrey Carlin**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T32	Kian-Tat Lim		
LVV-T374	John Swinbank		
LVV-T362	John Swinbank		
LVV-T368	John Swinbank		
LVV-T1756	Jeffrey Carlin		
LVV-T124	Simon Krughoff		
LVV-T199	Robert Gruendl [X]		
LVV-T83	Robert Lupton		
LVV-T190	Robert Gruendl [X]		
LVV-T77	Jim Bosch		
LVV-T84	Robert Lupton		
LVV-T88	Robert Lupton		
LVV-T115	Kian-Tat Lim		
LVV-T151	Colin Slater		
LVV-T1232	Colin Slater		
LVV-T149	Colin Slater		
LVV-T72	Jim Bosch		
LVV-T85	Robert Lupton		
LVV-T90	Robert Lupton		
LVV-T137	Colin Slater		
LVV-T55	Eric Bellm		
LVV-T146	Robert Gruendl [X]		
LVV-T66	Jim Bosch		
LVV-T91	Robert Lupton		
LVV-T126	Eric Bellm		
LVV-T1946	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1947	Jeffrey Carlin	Jeffrey Carlin	
LVV-T28	Colin Slater	Jeffrey Carlin	
LVV-T78	Kian-Tat Lim		
LVV-T42	Jim Bosch		
LVV-T38	Eric Bellm		
LVV-T141	Robert Gruendl [X]		

LVV-T140	Robert Gruendl [X]
LVV-T129	Robert Lupton
LVV-T127	Robert Lupton
LVV-T131	Gregory Dubois-Felsmann
LVV-T79	Jim Bosch
LVV-T1830	Jeffrey Carlin
LVV-T98	Kian-Tat Lim
LVV-T145	Robert Lupton
LVV-T144	Kian-Tat Lim
LVV-T74	Eric Bellm
LVV-T97	Kian-Tat Lim
LVV-T1529	Jeffrey Carlin
LVV-T1527	Jeffrey Carlin
LVV-T1758	Jeffrey Carlin
LVV-T1528	Jeffrey Carlin

4 Test Campaign Overview

4.1 Summary

T. Plan LVV-P99:	**Data Management Acceptance Test Campaign 1**			Draft
T. Cycle LVV-C208:	**Data Management Acceptance Test Campaign 1**			In Progress
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T32	1	Not Executed		
LVV-T374	1	Not Executed		
LVV-T362	1	Not Executed		
LVV-T368	2	Not Executed		
LVV-T1756	1	Not Executed		
LVV-T124	1	Not Executed		
LVV-T199	1	Not Executed		
LVV-T83	1	Not Executed		
LVV-T190	1	Not Executed		
LVV-T77	1	Not Executed		
LVV-T84	1	Not Executed		
LVV-T88	1	Not Executed		
LVV-T115	1	Not Executed		
LVV-T151	1	Not Executed		
LVV-T1232	1	Not Executed		
LVV-T149	1	Not Executed		
LVV-T72	1	Not Executed		
LVV-T85	1	Not Executed		
LVV-T90	1	Not Executed		
LVV-T137	1	Not Executed		
LVV-T55	1	Not Executed		
LVV-T146	1	Not Executed		
LVV-T66	1	Not Executed		
LVV-T91	1	Not Executed		
LVV-T126	1	Not Executed		

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

LVV-T78	1	Not Executed
LVV-T42	1	Not Executed
LVV-T38	1	Not Executed
LVV-T141	1	Not Executed
LVV-T140	1	Not Executed
LVV-T129	1	Not Executed
LVV-T127	1	Not Executed
LVV-T131	1	Not Executed
LVV-T79	1	Not Executed
LVV-T1830	1	Not Executed
LVV-T98	1	Not Executed
LVV-T145	1	Not Executed
LVV-T144	1	Not Executed
LVV-T74	1	Not Executed
LVV-T97	1	Not Executed
LVV-T1529	1	Not Executed
LVV-T1527	1	Not Executed
LVV-T1758	1	Not Executed
LVV-T1528	1	Not Executed

Table 2: Test Campaign Summary

4.2 Overall Assessment

Not yet available.

4.3 Recommended Improvements

Not yet available.

5 Detailed Test Results

5.1 Test Cycle LVV-C208

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

Test Cycle name: Data Management Acceptance Test Campaign 1
Status: In Progress

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

5.1.1 Software Version/Baseline

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

5.1.2 Configuration

Not provided.

5.1.3 Test Cases in LVV-C208 Test Cycle

5.1.3.1 LVV-T32 - Verify implementation of Raw Image Assembly

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Ingest data from the L1 Camera Test Stand DAQ.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Simulate all different modes of data gathering.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Verify that a raw image is constructed in correct format.

Expected Result

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

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Verify that a raw image is constructed with correct metadata.

– –

Expected Result

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

– –

Actual Result

5.1.3.2 LVV-T374 - Ingesting Camera test data

Version **1**. Open *LVV-T374* test case in Jira.

This test will check:

- That raw Camera test data is available on a filesystem in the LSST Data Facility;
- That raw Camera test data can be ingested and made available through the Data Management I/O abstraction (the “Data Butler”).

Preconditions:

Appropriate raw data from Camera test systems must be available on a filesystem within the LSST Data Facility. This test data is assumed to include visit 258334666 (hereafter referred to as $VISIT_ID) from RTM (Raft Tower Module) 007 for the purposes of this test, but other, equivalent, may be substituted.

At time of writing, suitable data may be found on the GPFS filesystem at /project/bootcamp/data/LCA-11021_RTM-007/7086/fe55_raft_acq/v0/44981. In future, as data transport procedures to the

Data Facility become more streamlined and formalised, this data may be moved elsewhere or made available through some other system. Throughout the test script, we use the string "$INPUT_DATA_DIR" as an alias for "/project/bootcamp/data/LCA-11021_RTM-007/7086/fe55_raft_acq/v0/44981" or wherever this data has been moved to.

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Connect to the Notebook Aspect of the Science Platform following the instructions at https://nb.lsst.io/. Log in, and "spawn" a new machine with image "Weekly 2018_45" and size "large".

Expected Result

The JupyterLab environment appears.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Create a terminal session. Use it to set up the LSST tools, then download and build version 5c12b06e6 of obs_lsst:

```
$ source /opt/lsst/software/stack/loadLSST.bash
$ setup lsst_distrib
$ git clone https://github.com/lsst/obs_lsst.git
$ cd obs_lsst
$ git checkout 5c12b06e6
$ setup -k -r .
```

$ scons

Expected Result

No errors are seen during execution of the provided commands.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Ingest RTM-007 test data by executing the following commands:

```
OUTPUT_REPO_DIR=$OUTPUT_DATA_DIR
INPUT_DATA_DIR=$INPUT_DATA_DIR
mkdir -p $OUTPUT_REPO_DIR
echo “lsst.obs.lsst.ts8.Ts8Mapper” > $OUTPUT_REPO_DIR/_mapper
ingestImages.py $OUTPUT_REPO_DIR $INPUT_DATA_DIR/*/*.fits
constructBias.py $OUTPUT_REPO_DIR –rerun calibs –id imageType=BIAS –batch-type smp –cores 4
ingestCalibs.py $OUTPUT_REPO_DIR –calibType bias $OUTPUT_REPO_DIR/rerun/calibs/bias/*/*.fits –validity 9999
–output $OUTPUT_REPO_DIR/CALIB –mode=link
```

Where:

$OUTPUT_DATA_DIR is some location on shared storage to which the user has write permission;
$INPUT_DATA_DIR is defined in the test case description.

Expected Result

Many status messages are logged to screen, and the command exits with status 0.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Demonstrate that raw and bias data for visit $VISIT_ID have been made available in the repository. Load a Python

interpreter (run “python”) and execute the following:

```
from lsst.daf.persistence import Butler
visit_id = $VISIT_ID
b = Butler($OUTPUT_DATA_DIR)
b.get(“raw”, visit=visit_id, detector=2)
b.get(“bias”, visit=visit_id, detector=2)
```

Expected Result

Each call to b.get() returns an instance of an ExposureF object. Warnings about lack of dark-time or WCS information may be ignored.

Actual Result

5.1.3.3 LVV-T362 - Installation of the LSST Science Pipelines Payloads

Version **1**. Open *LVV-T362* test case in Jira.

This test will check that:

- The Alert Production Pipeline payload is available for installation from documented channels;
- The Data Release Production Pipeline payload is available for installation from documented channels;
- The Calibration Products Production Pipeline payload is available for installation from documented channels;
- These payloads can be installed on systems at the LSST Data Facility following available documentation;
- The installed pipeline payloads are capable of successfully executing basic integration tests.

Note that this test assumes packaging of the Science Pipelines software, in which all the above

payloads are represented by a single “meta-package”, lsst_distrib.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

The LSST Science Pipelines, described by the lsst_distrib meta-package, should be installed following the documentation available at https://pipelines.lsst.io/. The suggested Conda environment will be used to ensure that a supported execution environment is available.

Expected Result

Detailed output will depend on the installation method chosen, but will confirm the successful installation of the Science Pipelines.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

The lsst_distrib top-level metapackage will be enabled. Assuming that the software has been installed at ${LSST_DIR}:

```
source ${LSST_DIR}/loadLSST.bash
setup lsst_distrib
```

Expected Result

Nothing is printed. The command

```
eups list -s lsst_distrib
```

may be used to confirm that the correct version of the codebase has been installed.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

The "LSST Stack Demo" package will be downloaded onto the test system from https://github.com/lsst/pipelines_check/releases. The version corresponding to to the version of the Science Pipelines under test should be chosen.

Expected Result

Depends on the tool selected by the user for downloading.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

The stack demo package is uncompressed into a directory ${DEMO_DIR}.

Expected Result

Depends on options given to the tar command. Should confirm the availability of the stack demo source.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

The demo package will be executed by following the instructions in its README file.

Expected Result

Successful execution will result in the string "Ok" being returned.

Actual Result

5.1.3.4 LVV-T368 - Loading and processing Camera test data

Version **2**. Open *LVV-T368* test case in Jira.

This test will check:

- That Camera test data is available for processing in the LSST Data Facility, and accessible through the LSST Science Platform;
- That the Data Management I/O abstraction (the "Data Butler") can load that data into the Science Platform environment;
- That Data Management algorithmic "tasks" can be executed to process that data;
- That results can be displayed in the Firefly display tool.

Preconditions:
Appropriate data — to include a "raw" and a "bias" exposure — from the Camera test systems must be available in a Butler data repository on a filesystem accessible to the Notebook Aspect of the Science Platform.

For the purposes of the following discussion, we assume that:

- Visit 258334666 from RTM (Raft Tower Module) 007 will be used;
- The data is available in a repository at /project/bootcamp/repo_RTM-007/ on the Data Facility GPFS filesystem

In the test script, we refer to “258334666” as “$VISIT_ID” and “/project/bootcamp/repo_RTM-007/” as “$REPOSITORY_PATH”; other data may be substituted as appropriate.

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Connect to the Notebook Aspect of the Science Platform following the instructions at https://nb.lsst.io/. Log in, and “spawn” a new machine with image “Weekly 2018_45“ and size “small”.

Expected Result

The JupyterLab environment appears.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Create a terminal session. Use it to set up the LSST tools, then download and build version 5c12b06e6 of obs_lsst:

```
$ source /opt/lsst/software/stack/loadLSST.bash
$ setup lsst_distrib
$ git clone https://github.com/lsst/obs_lsst.git
$ cd obs_lsst
$ git checkout 5c12b06e6
$ setup -k -r .
$ scons
```

Arrange for obs_lsst to automatically be added to the environment when starting a new notebook:

```
$ echo “setup -j -r ~/obs_lsst” >> ~/notebooks/.user_setups
```

Exit the terminal.

Expected Result

No errors are seen during execution of the provided commands.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Create a new “LSST” notebook.

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

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

and execute the cell.

Expected Result

Nothing is printed.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Create a Data Butler client, and use it to retrieve the data which will be used for this test.

```
butler = Butler($REPOSITORY_PATH)
```

```
raw = butler.get("raw", visit=$VISIT_ID, detector=2)
bias = butler.get("bias", visit=$VISIT_ID, detector=2)
```

Expected Result

Nothing is printed.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

Initialize the Firefly display system:

```
my_channel = '{}_test_channel'.format(os.environ['USER'])
server = 'https://lsst-lspdev.ncsa.illinois.edu'
ff='{}/firefly/slate.html?__wsch={}'.format(server, my_channel)
IFrame(ff,800,600)
afwDisplay.setDefaultBackend('firefly')
afw_display = afwDisplay.getDisplay(frame=1,
                                    name=my_channel)
```

Click on the link provided after executing the above.

Expected Result

A Firefly window is shown.

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

Display the raw image data in the Firefly window:

```
afw_display.mtv(raw)
```

Expected Result

Raw image data is displayed.

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

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

```
isr_config = IsrTask.ConfigClass()
isr_config.doDark=False
isr_config.doFlat=False
isr_config.doFringe=False
isr_config.doDefect=False
isr_config.doAddDistortionModel=False
isr_config.doLinearize=False
isr = IsrTask(config=isr_config)
result = isr.run(raw, bias=bias)
```

Expected Result

Nothing is printed.

Actual Result

Step 8 Step Execution Status: **Not Executed**

Description

Display the corrected image data in the Firefly window:

```
afw_display.mtv(result.exposure)
```

Expected Result

Processed (trimmed, bias-subtracted) image data is displayed.

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

Description

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

Example Code

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

Expected Result

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

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.6 LVV-T124 - Verify implementation of Software Architecture to Enable Community Re-Use

Version **1**. Open *LVV-T124* test case in Jira.

Show that the LSST software is capable of being executed in multiple contexts: single user instance, batch processing, continuous integration.
Also show that the algorithms can be reconfigured and, if desired, completely replaced at run time.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

Description

Using curated test datasets for multiple precursor instruments, verify and log that the prototype DRP pipelines execute successfully in three contexts:

1. The CI system
2. On a single user system: laptop, desktop, or notebook running in the Notebook aspect of the LSP.
3. Project workflow system.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Using a template testing notebook in the Notebook aspect of the LSP, verify and log the following:

1. Individual pipeline steps (tasks) are importable and executable on their own. this is not comprehensive, but demonstrative.
2. Individual pipeline steps may be overridden by configuration.
3. Users can implement a custom pipeline step and insert i into the processing flow via configuration.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

Read the resulting dataset using the Bulter, and confirm that it produced the desired data products.

Expected Result

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

Run subset of full DRP from previous step on an individual node. Was this organizationally easy? Did the performance scale appropriately?

Expected Result

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

Re-run aperture correction on subset. Verify that same results as DRP run are achieved.

Expected Result

Actual Result

Step 8 Step Execution Status: **Not Executed**

Description

Re-run photometric redshift estimation algorithm on subset coadd catalogs. Verify that same results are achieved as from full DRP.

Expected Result

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Not Executed**

Description
Analyze design

Expected Result

Actual Result

5.1.3.8 LVV-T83 - Verify implementation of Bad Pixel Map

Version **1**. Open *LVV-T83* test case in Jira.

Verify that the DMS can produce a map of detector pixels that suffer from pathologies, and that these pathologies are encoded in at least 32-bit values.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Interrogate the calibRegistry for the metadata associated with a bad pixel map, where the validity range contains the date of interest.

Expected Result

A bad pixel map for the requested date has been returned.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Check that the bad pixel pathologies are encoded as at least 32-bit values, and that the various pathologies are represented by different encoding.

Expected Result

Bad pixel values can be decoded to determine their pathologies using their 32-bit values.

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Not Executed**

Description

Analyze design

Expected Result

Actual Result

5.1.3.10 LVV-T77 - Verify implementation of Best Seeing Coadds

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Verify that these coadds exist.

Expected Result

Actual Result

5.1.3.11 LVV-T84 - Verify implementation of Bias Residual Image

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description
Identify the location of an appropriate precursor dataset.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result
Butler repo available for reading.

Actual Result

Step 3 Step Execution Status: **Not Executed**

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

Example Code
```
import osimport lsst.afw.display as afwDisplay
from lsst.daf.persistence import Butler
from lsst.ip.isr import IsrTask
from firefly_client import FireflyClient
from IPython.display import IFrame
```

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

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

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

Expected Result

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

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

Example Code

```
isr_config = IsrTask.ConfigClass()
isr_config.doDark=False
isr_config.doFlat=False
isr_config.doFringe=False
isr_config.doDefect=False
isr_config.doAddDistortionModel=False
isr_config.doLinearize=False
isr = IsrTask(config=isr_config)
result = isr.run(raw, bias=bias)
```

Expected Result

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

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.12 LVV-T88 - Verify implementation of Calibration Data Products

Version **1**. Open *LVV-T88* test case in Jira.

Verify that the DMS can produce and archive the required Calibration Data Products: cross talk correction, bias, dark, monochromatic dome flats, broad-band flats, fringe correction, and illumination corrections.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Identify a suitable set of calibration frames, including biases, dark frames, and flat-field frames.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Execute the Calibration Products Production payload. The payload uses raw calibration images and information from the Transformed EFD to generate a subset of Master Calibration Images and Calibration Database entries in the Data Backbone.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Confirm that the expected Master Calibration images and Calibration Database entries are present and well-formed.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Confirm that the expected data products are created, and that they have the expected properties.

Expected Result

A full set of calibration data products has been created, and they are well-formed.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

Test that the calibration products are archived, and can readily be applied to science data to produce the desired corrections.

Expected Result

Confirmation that application of the calibration products to processed data has the desired effects.

Actual Result

5.1.3.13 LVV-T115 - Verify implementation of Calibration Production Processing

Version **1**. Open *LVV-T115* test case in Jira.

Execute CPP on a variety of representative cadences, and verify that the calibration pipeline correctly produces necessary calibration products.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Identify a suitable set of calibration frames, including biases, dark frames, and flat-field frames.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Execute the Calibration Products Production payload. The payload uses raw calibration images and information from the Transformed EFD to generate a subset of Master Calibration Images and Calibration Database entries in the Data Backbone.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Confirm that the expected Master Calibration images and Calibration Database entries are present and well-formed.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Confirm that the expected data products are created, and that they have the expected properties.

Expected Result
Repos containing valid calibration products that are well-formed and ready to be applied to processed datasets.

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

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

Example Code

```
CELL 1:

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

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

CELL 2:

results = service.search("SELECT coord_ra, coord_dec, g_cModelFlux, r_cModelFlux \
                         FROM dp02_dc2_catalogs.Object \
                         WHERE CONTAINS(POINT('ICRS', coord_ra, coord_dec), \
                         CIRCLE('ICRS', 60.0, -30.0, 0.05)) = 1")
```

Expected Result

Screen output from CELL 1:

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

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

Table length=5533

coord_ra

coord_dec

g_cModelFlux

r_cModelFlux

deg

deg

nJy

nJy

float64

float64

float64

float64

59.9987401

-29.9728812

62.7060123

49.3496319

59.9995813

-29.9743232

166.0433743

394.8261645

59.9989853

-29.9750457

78.9557388

85.2691232

59.9993731

-29.9732406

111.0082072

165.6229656

60.0477786

-29.9736805

68.4818592

49.4783714

60.0400024

-29.9731507

52.0567337

114.2562171

60.0054666

-29.9728639

146.053072

134.1795803

60.00489

-29.9732239

1436.7150639

3606.8163133

60.0469583

-29.9735655

64.8838762

56.5677789

...

...

...

...

60.0053313

-30.0240394

125.6977786

379.8120713

59.9574061

-30.0163726

181.050889

200.8032979

60.0294415

-30.0241709

133.662163

230.8673464

59.9563419

-30.0239843

1551.2308712

4611.0406542

59.9879157

-30.0181116

76.3796313

46.5682713

60.0204061

-30.0228981

174.7738892

304.9991558

60.001638

-30.0183336

43.9593753

46.9695823

59.9861714

-30.0173405

164.6261404

288.8650875

59.9537443

-30.0160515

2228.7204658

5091.2041475

59.9683498

-30.0239539

835.415374

1101.0548649

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

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

Confirm that non-empty output files appear on disk.

Example Code

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

Expected Result

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

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

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

Check that these files contain the same number of rows:

Example Code

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


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

Expected Result

Print statement produces output "97058 97058 97058".

Actual Result

Step 8 Step Execution Status: **Not Executed**

Description

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

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

Expected Result

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

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

A credential-entry screen should be displayed.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

The Portal Aspect UI should be displayed following authentication.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Select query type “ADQL”.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

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

Example Code

```
SELECT cntr, ra, decl, w1mpro_ep, w2mpro_ep, w3mpro_ep FROM wise_00.allwise_p3as_mep WHERE CONTAINS(POINT('ICRS', ra, decl), CIRCLE('ICRS', 192.85, 27.13, .2)) = 1
```

Expected Result

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

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 8 Step Execution Status: **Not Executed**

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

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

Actual Result

5.1.3.16 LVV-T149 - Verify implementation of Catalog Queries

Version **1**. 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: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Example Code

SELECT * FROM Object WHERE qserv_areaspec_box(316.582327, −6.839078, 316.653938, −6.781822)

Expected Result

A catalog of objects satisfying the specified constraints.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Repeat the query from all available access routes (e.g., an external VO client, internal DM tools on the development cluster, 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

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

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

Verify the implementation of how Coadd images are created.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Verify that coadds were created following specification

Expected Result

Actual Result

5.1.3.18 LVV-T85 - Verify implementation of Crosstalk Correction Matrix

Version **1**. Open *LVV-T85* test case in Jira.

Verify that the DMS can generate a cross-talk correction matrix from appropriate calibration data.
Verify that the DMS can measure the effectiveness of the cross-talk correction matrix.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Identify an appropriate calibration dataset that can be used to derive the crosstalk correction matrix.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Execute the Calibration Products Production payload. The payload uses raw calibration images and information from the Transformed EFD to generate a subset of Master Calibration Images and Calibration Database entries in the Data Backbone.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Confirm that the expected Master Calibration images and Calibration Database entries are present and well-formed.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Confirm that the crosstalk correction matrix is produced and persisted.

Expected Result

A correction matrix quantifying what fraction of the signal detected in any given amplifier on each sensor in the focal plane appears in any other amplifier.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

Apply the crosstalk correction to simulated images, and confirm that the correction is performing as expected.

Expected Result

A noticeable difference between images before and after applying the correction.

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.20 LVV-T137 - Verify implementation of Data Product Ingest

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

Verify that data products can be ingested.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

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

Step 3 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.21 LVV-T55 - Verify implementation of DIAForcedSource Catalog

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Confirm that the DIAForcedSource catalog contains measurements for each source.

Expected Result

Actual Result

5.1.3.22 LVV-T146 - Verify implementation of DMS Initialization Component

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

Demonstrate that the DMS can be initialized in a safe state that will not allow data corruption/loss.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Power-cycle all of the DM systems at each Facility.

Expected Result

Restart of all DM systems.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Observe each system and ensure that it has recovered in a properly initialized state.

Expected Result

Systems are all active and initialized for their designated purpose.

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

5.1.3.24 LVV-T91 - Verify implementation of Fringe Correction Frame

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

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

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

Actual Result

Step 3 Step Execution Status: **Not Executed**

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

Expected Result
Images before and after correction have different pixel values.

Actual Result

5.1.3.25 LVV-T126 - Verify implementation of Image Differencing

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 4 Step Execution Status: **Not Executed**

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

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

Actual Result

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

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

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

Preconditions:

Execution status: **Pass**

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

This test case can be executed by running the script test_LVV-T1946.py, which is available in

the test report github repository's "scripts/" directory.

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

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

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

Example Code

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

Expected Result

Butler repo available for reading.

Actual Result

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

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

Step 2 Step Execution Status: **Pass**

Description

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

Expected Result

Actual Result

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

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

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

Step 3 Step Execution Status: **Pass**

Description

Verify that the coadd catalog provides measurements in flux units.

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

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

Screen outputs:
lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup
Input dataId: {'tract': 3828, 'band': 'i', 'patch': 13}
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count

base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count
base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_LocalBackground_instFlux ..... count
base_LocalBackground_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_0_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_0_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_0_7_instFlux ..... count
ext_gaap_GaapFlux_1_15x_0_7_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_1_0_instFlux ..... count
ext_gaap_GaapFlux_1_15x_1_0_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_1_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_1_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_2_5_instFlux ..... count
ext_gaap_GaapFlux_1_15x_2_5_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_3_0_instFlux ..... count
ext_gaap_GaapFlux_1_15x_3_0_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_PsfFlux_instFlux ..... count
ext_gaap_GaapFlux_1_15x_PsfFlux_instFluxErr ..... count
ext_gaap_GaapFlux_1_15x_Optimal_instFlux ..... count
ext_gaap_GaapFlux_1_15x_Optimal_instFluxErr ..... count
ext_photometryKron_KronFlux_instFlux ..... count
ext_photometryKron_KronFlux_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_0_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_0_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_0_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_0_6_0_instFluxErr ..... count

ext_convolved_ConvolvedFlux_0_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_0_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_1_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_1_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_1_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_1_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_1_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_2_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_2_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_2_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_2_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_2_kron_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_3_3_instFlux ..... count
ext_convolved_ConvolvedFlux_3_3_3_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_4_5_instFlux ..... count
ext_convolved_ConvolvedFlux_3_4_5_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_6_0_instFlux ..... count
ext_convolved_ConvolvedFlux_3_6_0_instFluxErr ..... count
ext_convolved_ConvolvedFlux_3_kron_instFlux ..... count
ext_convolved_ConvolvedFlux_3_kron_instFluxErr ..... count
modelfit_CModel_initial_instFlux ..... count
modelfit_CModel_initial_instFluxErr ..... count
modelfit_CModel_initial_instFlux_inner ..... count
modelfit_CModel_exp_instFlux ..... count
modelfit_CModel_exp_instFluxErr ..... count
modelfit_CModel_exp_instFlux_inner ..... count
modelfit_CModel_dev_instFlux ..... count
modelfit_CModel_dev_instFluxErr ..... count
modelfit_CModel_dev_instFlux_inner ..... count
modelfit_CModel_instFlux ..... count
modelfit_CModel_instFluxErr ..... count
modelfit_CModel_instFlux_inner ..... count
undeblended_base_CircularApertureFlux_3_0_instFlux ..... count
undeblended_base_CircularApertureFlux_3_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_4_5_instFlux ..... count
undeblended_base_CircularApertureFlux_4_5_instFluxErr ..... count
undeblended_base_CircularApertureFlux_6_0_instFlux ..... count
undeblended_base_CircularApertureFlux_6_0_instFluxErr ..... count

undeblended_base_CircularApertureFlux_9_0_instFlux ..... count
undeblended_base_CircularApertureFlux_9_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_12_0_instFlux ..... count
undeblended_base_CircularApertureFlux_12_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_17_0_instFlux ..... count
undeblended_base_CircularApertureFlux_17_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_25_0_instFlux ..... count
undeblended_base_CircularApertureFlux_25_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_35_0_instFlux ..... count
undeblended_base_CircularApertureFlux_35_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_50_0_instFlux ..... count
undeblended_base_CircularApertureFlux_50_0_instFluxErr ..... count
undeblended_base_CircularApertureFlux_70_0_instFlux ..... count
undeblended_base_CircularApertureFlux_70_0_instFluxErr ..... count
undeblended_base_PsfFlux_instFlux ..... count
undeblended_base_PsfFlux_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_5_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_7_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_0_7_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_0_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_0_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_5_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_1_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_2_5_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_2_5_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_3_0_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_3_0_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_PsfFlux_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_PsfFlux_instFluxErr ..... count
undeblended_ext_gaap_GaapFlux_1_15x_Optimal_instFlux ..... count
undeblended_ext_gaap_GaapFlux_1_15x_Optimal_instFluxErr ..... count
undeblended_ext_photometryKron_KronFlux_instFlux ..... count
undeblended_ext_photometryKron_KronFlux_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_0_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_0_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_3_3_instFluxErr ..... count

undeblended_ext_convolved_ConvolvedFlux_1_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_1_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_1_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_2_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_2_kron_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_3_3_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_3_3_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_4_5_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_4_5_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_6_0_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_6_0_instFluxErr ..... count
undeblended_ext_convolved_ConvolvedFlux_3_kron_instFlux ..... count
undeblended_ext_convolved_ConvolvedFlux_3_kron_instFluxErr ..... count

All forced_src instFlux entries have units of counts: True

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

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

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

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

Preconditions:

Execution status: **Pass**

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

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

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

Detailed steps results:

Step 1 Step Execution Status: **Pass**

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

Example Code

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

Expected Result
Butler repo available for reading.

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

For DP0.2 data on the IDF:

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

Step 2 Step Execution Status: **Pass**

Description

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

Expected Result

Actual Result

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

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

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

Step 3 Step Execution Status: **Pass**

Description

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

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

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

Screen outputs:

```
lsst_distrib        g0b29ad24fb+cafeaf151e    current w_2022_32 setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count
base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count
base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_PeakLikelihoodFlux_instFlux ..... count
base_PeakLikelihoodFlux_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ip_diffim_NaiveDipoleFlux_pos_instFlux ..... count
ip_diffim_NaiveDipoleFlux_pos_instFluxErr ..... count
ip_diffim_NaiveDipoleFlux_neg_instFlux ..... count
ip_diffim_NaiveDipoleFlux_neg_instFluxErr ..... count
ip_diffim_PsfDipoleFlux_pos_instFlux ..... count
ip_diffim_PsfDipoleFlux_pos_instFluxErr ..... count
ip_diffim_PsfDipoleFlux_neg_instFlux ..... count
ip_diffim_PsfDipoleFlux_neg_instFluxErr ..... count
ip_diffim_DipoleFit_pos_instFlux ..... count
ip_diffim_DipoleFit_pos_instFluxErr ..... count
ip_diffim_DipoleFit_neg_instFlux ..... count
ip_diffim_DipoleFit_neg_instFluxErr ..... count
ip_diffim_DipoleFit_instFlux ..... count
ip_diffim_forced_PsfFlux_instFlux ..... count
```

ip_diffim_forced_PsfFlux_instFluxErr ..... count

All diaSrc table instFlux entries have units of counts: True

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

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

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

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

Preconditions:

Execution status: **Pass**

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

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

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

Detailed steps results:

Step 1	Step Execution Status: **Pass**

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

Example Code

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

Expected Result
Butler repo available for reading.

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

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

Step 2 Step Execution Status: **Pass**

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

Expected Result

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

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

visit = 924041

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

Step 3 Step Execution Status: **Pass**

Description

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

Expected Result

Confirmation of measurements in catalogs encoded in flux units.

Actual Result

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

lsst_distrib g0b29ad24fb+cafeaf151e current w_2022_32 setup
Input dataId: {'instrument': 'LSSTCam-imSim', 'visit': 924041, 'detector': 5}
deblend_psf_instFlux ..... count
base_Blendedness_raw_child_instFlux ..... count
base_Blendedness_raw_parent_instFlux ..... count
base_Blendedness_abs_child_instFlux ..... count
base_Blendedness_abs_parent_instFlux ..... count
base_SdssShape_instFlux ..... count
base_SdssShape_instFluxErr ..... count
base_CircularApertureFlux_3_0_instFlux ..... count
base_CircularApertureFlux_3_0_instFluxErr ..... count
base_CircularApertureFlux_4_5_instFlux ..... count
base_CircularApertureFlux_4_5_instFluxErr ..... count
base_CircularApertureFlux_6_0_instFlux ..... count
base_CircularApertureFlux_6_0_instFluxErr ..... count
base_CircularApertureFlux_9_0_instFlux ..... count
base_CircularApertureFlux_9_0_instFluxErr ..... count
base_CircularApertureFlux_12_0_instFlux ..... count
base_CircularApertureFlux_12_0_instFluxErr ..... count
base_CircularApertureFlux_17_0_instFlux ..... count
base_CircularApertureFlux_17_0_instFluxErr ..... count
base_CircularApertureFlux_25_0_instFlux ..... count
base_CircularApertureFlux_25_0_instFluxErr ..... count

base_CircularApertureFlux_35_0_instFlux ..... count
base_CircularApertureFlux_35_0_instFluxErr ..... count
base_CircularApertureFlux_50_0_instFlux ..... count
base_CircularApertureFlux_50_0_instFluxErr ..... count
base_CircularApertureFlux_70_0_instFlux ..... count
base_CircularApertureFlux_70_0_instFluxErr ..... count
base_GaussianFlux_instFlux ..... count
base_GaussianFlux_instFluxErr ..... count
base_LocalBackground_instFlux ..... count
base_LocalBackground_instFluxErr ..... count
base_PsfFlux_instFlux ..... count
base_PsfFlux_instFluxErr ..... count
ext_photometryKron_KronFlux_instFlux ..... count
ext_photometryKron_KronFlux_instFluxErr ..... count

All src table instFlux entries have units of counts: True

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

5.1.3.29 LVV-T78 - Verify implementation of Persisting Data Products

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

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

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

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Produce some relevant coadds and store them in the Archive

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Examine the data retention policies for those products

Expected Result

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Ingest the data from an appropriate processed dataset.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Inspect the calexp image to ensure that

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

Expected Result

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

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

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

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

Actual Result

Step 7 Step Execution Status: **Not Executed**

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

Expected Result

Actual Result

5.1.3.31 LVV-T38 - Verify implementation of Processed Visit Images

Version **1**. Open *LVV-T38* test case in Jira.

Verify that the DMS
1. Successfully produces Processed Visit Images, where the instrument signature has been removed.
2. Successfully combines images obtained during a standard visit.

The verification should include confirming that the images have been trimmed of the over-scan, and that correction of the instrumental signature (including crosstalk) has been applied properly.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Identify suitable precursor datasets containing unprocessed raw images.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Run the initial steps (including instrument signature removal and calibration) of Data Release (or Prompt) Processing on these data. Verify that Processed Visit Images are generated at the correct size and with significant instrumental artifacts removed.

Expected Result

Raw precursor dataset images have been processed into Processed Visit Images, with instrumental artifacts corrected.

Actual Result

5.1.3.32 LVV-T141 - Verify implementation of Production Monitoring

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

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

Preconditions:

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

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.33 LVV-T140 - Verify implementation of Production Orchestration

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

Demonstrate use to orchestration software to perform real-time and batch production on LSST compute platform(s).

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Identify an appropriate precursor dataset.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.34 LVV-T129 - Verify implementation of Provide Calibrated Photometry

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Ingest the data products from an appropriate AP processing dataset.

Expected Result

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

5.1.3.36 LVV-T131 - Verify implementation of Provide User Interface Services

Version **1**. Open *LVV-T131* test case in Jira.

Verify the availability and functionality of the broad range of user interface services called for in the requirement, as applied to both Nightly and DRP data. This will primarily be done by verifications performed at the LSST Science Platform level, based on the requirements in LDM-554; however, a high-level set of tests corresponding to the DMS-REQ-0160 requirement are defined below.

Preconditions:

1. Testing this requirement relies on a set of data products meeting the data model implied by the DPDD existing in a deployment of the Science Platform and its underlying database and file services.
   (a) In particular, both image and catalog data products are required.
   (b) From the specific language of the underlying requirement, it appears clear that coadded data products are required, but in practice single-epoch data products should be included in the test as well.
2. Depending on when this requirement is tested, the tests may involve either or both of precursor data and LSST commissioning data. The use of the latter is ultimately essential to ensure that the tests are performed with as LSST-like a dataset as possible.

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Establishment of test coordinates:

Establish sky positions and surrounding regions (e.g., cones or polygons), field sizes, filter bands, and temporal epochs for the tests that are consistent with the known content of the test dataset, whether precursor or LSST commissioning data.

Establishing sky positions should include pre-determining the corresponding LSST "tract and patch" identifiers.

If the plan to not keep all calibrated single-epoch images on disk is still in place at the time of the test, identify for use in the test both images that are, and are not, on disk.

Establish target image boundaries, projections, and pixel scales to be used for resampling tests. Ensure that at least some of these test conditions include coadded image boundaries that cross tract and patch boundaries, and single-epoch image boundaries that cross focal plane raft boundaries.

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Butler image access:

From within the Notebook Aspect, verify that coadded images for the identified regions of sky and filter bands are accessible via the Butler. Verify that the same images are available whether obtained by direct reference to the previous established tract/patch identifiers or by the use of LSST stack code for retrieving images based on sky coordinates.

From within the Notebook Aspect, verify that single-epoch raw images for the selected locations and times are available. Verify that calibrated images (PVIs) for the selected locations and times are available; depending on the details of the test dataset, verify that PVIs still on disk can be retrieved immediately.

Verify that lists or tables of image metadata, not just individual images, can be retrieved. E.g., a list of all the single-epoch images covering a selected sky location.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Programmatic PVI re-creation:

From within the Notebook Aspect, verify that the recreation on demand of a PVI can be performed. Ideally, this should be done as follows:

- Verify that recreation of a PVI that *is* still available works and that it reproduces the original PVI exactly (except for provenance metadata that must be different) or within the reasonable ability of processing systems to do so (e.g., taking into account that the original calibration and the recreation may have run on different CPU architectures).
- The test conditions should ensure the verification that a recreation was actually performed, i.e., that the still-available PVI was not returned instead.
- Note that it does not appear to be a requirement that *at Butler level* recreation on demand of PVIs is a completely transparent process. If this *is* decided to be a requirement, the test must also verify that it has been satisfied. If it is *not* a requirement, verify that adequate documentation on the PVI-recreation process (e.g., the SuperTasks and configuration to be used) is available.

Expected Result

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Butler catalog access:

From within the Notebook Aspect, verify that all the catalog data products described in the DPDD can be retrieved for the coordinates selected above via the Butler. (This test should include access to SSObject data, but the details of how such a test would depend on the coordinate selections require additional thought.)

Expected Result

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

LSST-stack-based resampling/reprojection:

Verify the availability of software in the LSST stack, and associated documentation, that permits the resampling of LSST images to different pixel grids and projections.

Exercise this capability for the test conditions selected in Step 1 above.

Perform photometric and astrometric tests on the resulting resampled images to provide evidence that the transformations performed were correct to the accuracy supported by the data.

Expected Result

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

Comment:

The following API Aspect test steps should be carried out on the required "offsite-like" test platform, to ensure that their success does not reflect any privileged access given to processes inside the Data Access Center or other Science Platform instance. However, at least a small sampling of them should *also* be carried out *within* the Science Platform environment, i.e., in the Notebook Aspect, and the results compared.

Expected Result

Actual Result

Step 7 Step Execution Status: **Not Executed**

Description

API Aspect image access:

Using IVOA services such as the Registry and ObsTAP, from the "offsite-like" test platform, verify that the existence of the classes of image data products foreseen in the DPDD can be determined.

Verify that ObsTAP and/or SIAv2 can be used to find the same images and lists of images for the established test

coordinates that were retrieved via the Butler in Step 2 above.
Verify that the selected images are retrievable from the Web services.
Verify that the retrieved images are identical in their pixel content and metadata.
The tests must include both coadded and single-epoch images.

Expected Result

Actual Result

Step 8 Step Execution Status: **Not Executed**

Description

API Aspect image transformations:
Verify that image cutouts and resamplings can be performed via the IVOA SODA service, and that the results are identical to those obtained for the same parameters from the LSST-stack-based tests in Step 5.
(The requirements for supported reprojections, if any, in the SODA service have not been established at the time of writing.)

Expected Result

Actual Result

Step 9 Step Execution Status: **Not Executed**

Description

API Aspect catalog data access:
Verify that the IVOA Registry, RegTAP, TAP_SCHEMA, and other relevant mechanisms can be used to discover the existence of all the catalog data products foreseen in the DPDD.
Using the IVOA TAP service, verify that all the catalog data products foreseen in the DPDD can be retrieved for the coordinates determined in Step 1. Verify that their scientific content is the same as when they are retrieved via the Butler.

Expected Result

Actual Result

Step 10 Step Execution Status: **Not Executed**

Description

Comment:

The Portal Aspect tests below should be carried out from a web browser on an "offsite-like" test platform, to ensure that no privileged access provided to intra-data-center clients is relied upon.

Expected Result

Actual Result

Step 11 Step Execution Status: **Not Executed**

Description

Portal Aspect data browsing:

Verify that the Portal Aspect can be used to discover the existence of all the data products foreseen in the DPDD.
Verify that the UI permits locating the data for the coordinates selected in Step 1 by visual means, e.g., by zooming and panning in from an all-sky view.
Verify that the UI permits locating the data by typing in coordinates as well.

Expected Result

Actual Result

Step 12 Step Execution Status: **Not Executed**

Description

Portal Aspect image access:

Verify that the Portal Aspect allows both the retrieval of "original" image data, i.e., in its native LSST pixel projection and with full metadata, as well as retrieval of on-demand UI cutouts of coadded image data for selected locations.

Expected Result

Actual Result

Step 13 Step Execution Status: **Not Executed**

Description

Portal Aspect catalog query and visualization:

Verify that the Portal Aspect allows graphical querying of DPDD catalog data, both coadded and single-epoch, for selected regions of sky and/or with selected properties, and supports the visualization of the results (including histogramming, scatterplots, time series, table manipulations, and overplotting on image data).
(Note that the Science Platform requirements, LDM-554, lay out a detailed set of requirements on the selection and visualization of catalog data.)

Expected Result

Actual Result

Step 14 Step Execution Status: **Not Executed**

Description

Portal Aspect data download:

Verify that data identified and/or visualized in the Portal Aspect can be downloaded to the remote system running the web browser in which the Portal is displayed, as well as to the User Workspace.

Expected Result

Actual Result

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

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

Verify that the DRP pipelines produce PSF matched coadds.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

Description

Verify that PSF-matched coadds were created.

Expected Result

Actual Result

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

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Image with bright stars is displayed.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Confirm that each of the following is true:

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

Expected Result

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

Actual Result

5.1.3.39 LVV-T98 - Verify implementation of Selection of Datasets

Version **1**. Open *LVV-T98* test case in Jira.

Verify that the DMS can identify and retrieve datasets consisting of logical groupings of Exposures, metadata, provenance, etc., or other groupings that are processed or produced as a logical unit.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

Ingest data from an appropriate processed dataset.

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Observe retrieval of single Processed Visit Image (PVI) with metadata.

Expected Result

A PVI and its associated metadata.

Actual Result

Step 4 Step Execution Status: **Not Executed**

Description

Observe retrieval of multiple PVIs with metadata.

Expected Result

A set of PVIs and their associated metadata.

Actual Result

Step 5 Step Execution Status: **Not Executed**

Description

Observe retrieval of coadd patch with metadata and provenance information.

Expected Result

An image of coadded data in a patch, along with its metadata and information describing the provenance of the patch constituents.

Actual Result

Step 6 Step Execution Status: **Not Executed**

Description

Observe retrieval of subset of rows in each of the above catalogs.

Expected Result

Actual Result

5.1.3.40 LVV-T145 - Verify implementation of Task Configuration

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

5.1.3.41 LVV-T144 - Verify implementation of Task Specification

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description
Verify that an example science algorithm can be run through one of these Tasks. Three examples from different areas: source measurement, image subtraction, and photometric-redshift estimation.

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

Actual Result

5.1.3.42 LVV-T74 - Verify implementation of Template Coadds

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

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

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

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

Actual Result

Step 2 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 3 Step Execution Status: **Not Executed**

Description

Confirm that the template coadds have been created and are well-formed.

Expected Result

Actual Result

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

Version **1**. 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: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

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

Expected Result

Actual Result

Step 2 Step Execution Status: **Not Executed**

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

Step 3 Step Execution Status: **Not Executed**

Description

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

Example Code

```
import lsst.daf.persistence as dafPersist
butler = dafPersist.Butler(inputs='DATA/path')
```

Expected Result

Butler repo available for reading.

Actual Result

Step 4 Step Execution Status: **Not Executed**

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

Step 5 Step Execution Status: **Not Executed**

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

5.1.3.44 LVV-T1529 - Verify Production of All-Sky HiPS Map

Version **1**. Open *LVV-T1529* test case in Jira.

Verify that Data Release Production includes the production of an all-sky image map for the existing coadded image area in each filter band, and at least one pre-defined all-sky color image map, following the IVOA HiPS Recommendation.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Expected Result

Actual Result

5.1.3.45 LVV-T1527 - Verify Support for HiPS Visualization

Version **1**. Open *LVV-T1527* test case in Jira.

Verify that the LSST Science Platform supports the visualization of LSST-generated HiPS image maps as well as other HiPS maps which satisfy the IVOA HiPS Recommendation. Also verify that integrated behavior is available, such as the overplotting of catalog entries, comparable to that provided for individual source images (e.g., PVIs and coadd tiles).

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Expected Result

Actual Result

5.1.3.46 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**. 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 to computed values of the PF1 metric.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

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

Step 2 Step Execution Status: **Not Executed**

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

Step 3 Step Execution Status: **Not Executed**

Description

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

Example Code

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

Expected Result

The output collection (in this case, "u/username/faro_metrics") containing metric measurements and any associ-

ated extras and metadata is available via the butler.

Actual Result

Step 4 Step Execution Status: **Not Executed**

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 PA2uzy has been calculated (and that it used PF1).

Actual Result

5.1.3.47 LVV-T1528 - Verify Visualization of MOCs via Science Platform

Version **1**. Open *LVV-T1528* test case in Jira.

Verify that the LSST Science Platform supports the visualization of the LSST-generated MOCs as well as other MOCs which satisfy the IVOA MOC Recommendation.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Not Executed**

Description

Expected Result

Actual Result

A Documentation

The verification process is defined in LSE-160. The use of Docsteady to format Jira information in various test and planing documents is described in DMTN-140 and practical commands are given in DMTN-178.

B Acronyms used in this document

Acronym	Description
ADQL	Astronomical Data Query Language
AP	Alert Production
API	Application Programming Interface
CI	Continuous Integration
CPP	Calibration Production Processing
CPU	Central Processing Unit
CSV	Comma Separated Values
ComCam	The commissioning camera is a single-raft, 9-CCD camera that will be installed in LSST during commissioning, before the final camera is ready.
DAQ	Data Acquisition System
DC2	Data Challenge 2 (DESC)
DESC	Dark Energy Science Collaboration
DIA	Difference Image Analysis
DM	Data Management
DMS	Data Management Subsystem
DMS-REQ	Data Management System Requirements prefix
DMSR	DM System Requirements; LSE-61
DMTN	DM Technical Note
DP0	Data Preview 0
DPDD	Data Product Definition Document
DRP	Data Release Production
DS9	Deep Space 9 (specific astronomical data visualisation application; SAOImage)
EFD	Engineering and Facility Database
FITS	Flexible Image Transport System

GPFS	General Parallel File System (now IBM Spectrum Scale)
HSC	Hyper Suprime-Cam
IDF	Interim Data Facility
IPAC	No longer an acronym; science and data center at Caltech
ISR	Instrument Signal Removal
IVOA	International Virtual-Observatory Alliance
JSON	JavaScript Object Notation
L1	Lens 1
LCA	Document handle LSST camera subsystem controlled documents
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
MOC	Multi Ordered Catalogue
PDR	Preliminary Design Review
PMCS	Project Management Controls System
PSF	Point Spread Function
PVI	Processed Visit Image
QA	Quality Assurance
QSERV	LSST Query Services
RA	Right Ascension
RMS	Root-Mean-Square
RSP	Rubin Science Platform
RTM	Raft Tower Module
SODA	Server-side Operations for Data Access
SQL	Structured Query Language
TAP	Table Access Protocol
TOPCAT	Tool for OPerations on Catalogues And Tables
UI	User Interface
US	United States
USDF	United States Data Facility
VO	Virtual Observatory

WCS	World Coordinate System
deg	degree; unit of angle