Vera C. Rubin Observatory
Software Test Report

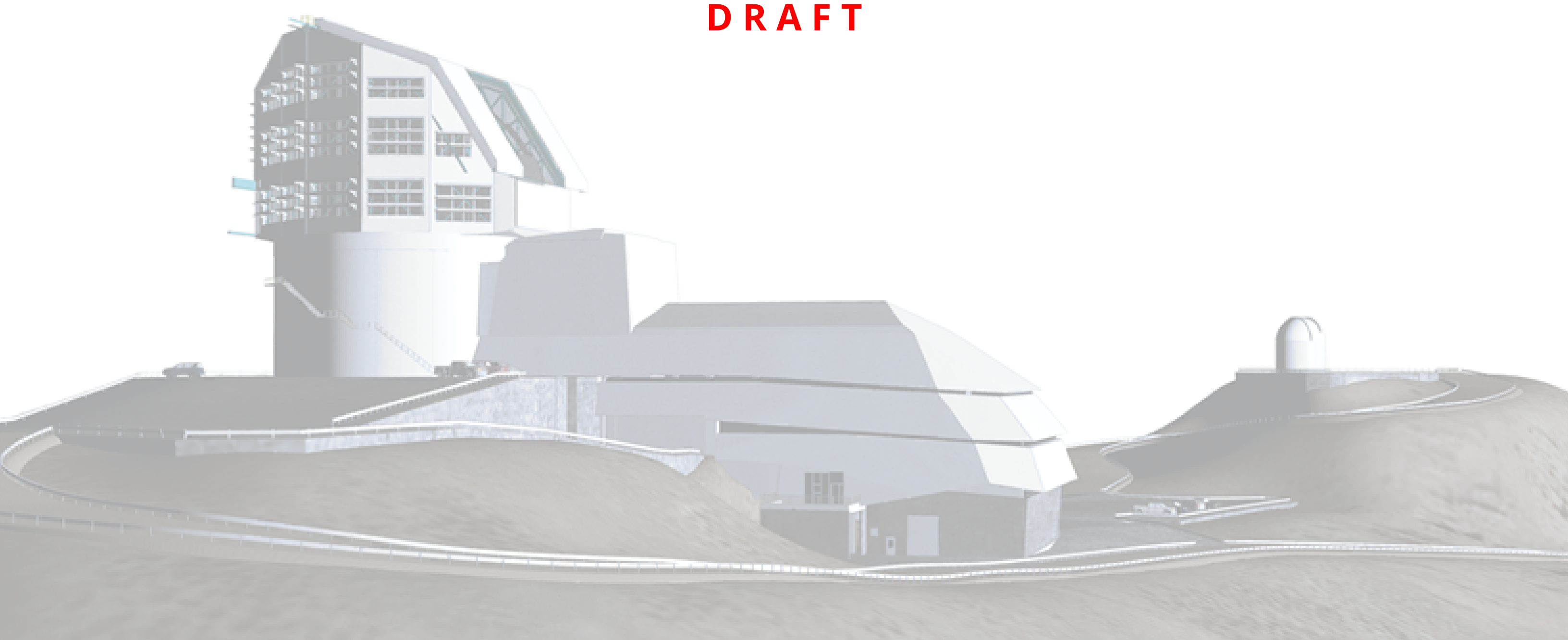
LDM-GEN3: Gen 3 Butler Acceptance Testing Test Plan and Report

Jeff Carlin, Robert Gruendl, Leanne Guy

DMTR-271

Latest Revision: 2022-05-13

DRAFT



Abstract

This is the test plan and report for **Gen 3 Butler Acceptance Testing** (LDM-GEN3), an LSST milestone pertaining to the Data Management Subsystem.
This document is based on content automatically extracted from the Jira test database on 2022-05-13. The most recent change to the document repository was on 2022-05-13.

Change Record

Version	Date	Description	Owner name
	2020-10-30	First draft	Robert Gruendl
	2021-01-28	Include new test cycle for LDM-556 requirements	Leanne Guy

Document curator: Jeff Carlin
Document source location: `https://github.com/lsst-dm/DMTR-271`
Version from source repository: 1d535aa

Contents

LDM-GEN3: Gen 3 Butler Acceptance Testing Test Plan and Report

1 Introduction

1.1 Objectives

The goal of this test is to demonstrate that the Gen3 Butler software project has sufficiently matured that subsequent DM development should begin focusing on adoption of Gen3 Butler software are repositories throughout the DM software project (i.e. that deprecation of Gen2 Butler usage within the project can begin).

1.2 System Overview

The Gen3 refactoring of the Butler is central to evolution of the overall DM software design and has repercussions throughout the rest of the DM project. This test plan is designed to verify that minimal requirements have been met and the DM project can now begin the process of integrating the Gen3 Butler within the pipelines and analysis tools. Those minimal requirements are that:

1. possible to ingest raw dataset types central to the Rubin operations and the ongoing development of the data management systems..
2. cp_pipe equivalent under Gen3 is available
3. developers can run a pipeline with a single-node using pipetask
4. processing supporting development is possible in a reasonable time (e.g. a 3-tract RC2 test run can be accomplished within a reasonable time)
5. Calibration Product Pipelines (CPP) can be run to support above investigations
6. Batch Processing System (BPS) is available to support testing at larger scales

In addition, at the time these tests occur the Gen3 Butler schema be considered stable enough that changes no longer occur on a weekly basis (i.e forced re-ingestion/migration of existing repositories are no longer a weekly occurrence). Changes requiring wholesale reingestion/migration may still be required but will occur in a regimented manner and the choice to allow schema changes without an accompanying means to migrate old repositories would become a change-control board (CCB) level issue.

Applicable Documents:
LDM-592: Data Access Use Cases
LDM-556: Data Management Middleware Requirements
LDM-639: Data Management Acceptance Test Specification

1.3 Document Overview

This document was generated from Jira, obtaining the relevant information from the LVV-P77 Jira Test Plan and related Test Cycles (LVV-C160 LVV-C162 LVV-C190).

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

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

The current status of test plan LVV-P77 in Jira is **Approved** .

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/`

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

[3] **[LDM-556]**, Dubois-Felsmann, G., Jenness, T., Bosch, J., et al., 2018, *Data Management Middleware Requirements*, LDM-556, URL https://ldm-556.lsst.io/

[4] **[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/

[5] **[LDM-592]**, Jenness, T., Bosch, J., Gower, M., et al., 2018, *Data Access Use Cases*, LDM-592, URL https://ldm-592.lsst.io/

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

2 Test Plan Details

2.1 Data Collection

Observing is not required for this test campaign.

2.2 Verification Environment

These tests assume a stable weekly stack which supports Gen3 running of the above, that services that automatically ingest new data can support on-going ingestion to Gen3 repositories (i.e. DBB shared spaces and OODS support serving data through Gen3), and that batch processing services can support pipeline execution of Gen3 products.

2.3 Related Documentation

Jira Attachments

To LVV-C160 results	DM-Gen2MiddlewareRemovalPlanning-080222-2302-1360.pdf
To LVV-C160 results	DRP.yaml
To LVV-C160 results	DRP-RC2.yaml
To LVV-C160 results	DRP-ci_hsc+fakes.yaml
To LVV-C160 results	pipeline_detail.png
To LVV-C160 results	pipeline.png
To LVV-C160 results	ci_hsc_log_w_2022_05.log

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

2.4 PMCS Activity

Primavera milestones related to the test campaign:

- LDM-503-GEN3-01

3 Personnel

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

	T. Plan LVV-P77 owner:	**Jeffrey Carlin**	
	T. Cycle LVV-C160 owner:	**Jeffrey Carlin**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T2264	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1984	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1982	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1987	Jeffrey Carlin	Jeffrey Carlin	
LVV-T1983	Jeffrey Carlin	Jeffrey Carlin	
	T. Cycle LVV-C162 owner:	**Leanne Guy**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T1985	Leanne Guy	Leanne Guy	
	T. Cycle LVV-C190 owner:	**Jeffrey Carlin**	
Test Cases	**Assigned to**	**Executed by**	**Additional Test Personnel**
LVV-T2503	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2502	Leanne Guy		
LVV-T2501	Leanne Guy		
LVV-T2499	Leanne Guy		
LVV-T2498	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2497	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2496	Leanne Guy		
LVV-T2495	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2494	Leanne Guy	Jeffrey Carlin	
LVV-T2493	Leanne Guy	Jeffrey Carlin	
LVV-T2492	Leanne Guy	Jeffrey Carlin	
LVV-T2491	Leanne Guy	Jeffrey Carlin	
LVV-T2488	Leanne Guy	Jeffrey Carlin	
LVV-T2487	Leanne Guy		
LVV-T2486	Leanne Guy		
LVV-T2485	Leanne Guy		
LVV-T2484	Leanne Guy		
LVV-T2483	Leanne Guy		
LVV-T2482	Leanne Guy	Jeffrey Carlin	
LVV-T2481	Leanne Guy	Jeffrey Carlin	
LVV-T2480	Jeffrey Carlin	Jeffrey Carlin	
LVV-T2479	Jeffrey Carlin	Jeffrey Carlin	

LVV-T2478	Leanne Guy	
LVV-T2477	Leanne Guy	
LVV-T2476	Leanne Guy	
LVV-T2474	Leanne Guy	Jeffrey Carlin
LVV-T2475	Leanne Guy	Jeffrey Carlin
LVV-T2473	Leanne Guy	Jeffrey Carlin
LVV-T2472	Leanne Guy	
LVV-T2471	Jeffrey Carlin	Jeffrey Carlin
LVV-T2470	Leanne Guy	Jeffrey Carlin
LVV-T2469	Leanne Guy	Jeffrey Carlin
LVV-T2468	Leanne Guy	Jeffrey Carlin
LVV-T2466	Jeffrey Carlin	
LVV-T2467	Leanne Guy	Jeffrey Carlin
LVV-T2464	Leanne Guy	Jeffrey Carlin
LVV-T2465	Jeffrey Carlin	
LVV-T2461	Leanne Guy	
LVV-T2463	Jeffrey Carlin	
LVV-T2462	Jeffrey Carlin	Jeffrey Carlin
LVV-T2460	Jeffrey Carlin	Jeffrey Carlin
LVV-T2457	Jeffrey Carlin	
LVV-T2456	Jeffrey Carlin	Jeffrey Carlin
LVV-T2455	Jeffrey Carlin	
LVV-T2454	Jeffrey Carlin	
LVV-T2458	Jeffrey Carlin	Jeffrey Carlin
LVV-T2451	Jeffrey Carlin	Jeffrey Carlin
LVV-T2453	Jeffrey Carlin	Jeffrey Carlin
LVV-T2449	Jeffrey Carlin	Jeffrey Carlin
LVV-T2452	Jeffrey Carlin	Jeffrey Carlin
LVV-T2450	Jeffrey Carlin	
LVV-T2447	Leanne Guy	
LVV-T2446	Leanne Guy	Jeffrey Carlin
LVV-T2444	Leanne Guy	Jeffrey Carlin
LVV-T2442	Leanne Guy	Jeffrey Carlin
LVV-T2443	Leanne Guy	Jeffrey Carlin
LVV-T2441	Leanne Guy	
LVV-T2440	Leanne Guy	Jeffrey Carlin
LVV-T2439	Leanne Guy	Jeffrey Carlin

4 Test Campaign Overview

4.1 Summary

T. Plan LVV-P77:		**LDM-GEN3: Gen 3 Butler Acceptance Testing**		Approved
T. Cycle LVV-C160:		**LDM-503-GEN3: Gen 3 Butler Acceptance Testing**		Done
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T2264	1	Pass		
LVV-T1984	1	Pass		
LVV-T1982	1	Pass	Working on lsst-devl02, in directory /project/jcarlin/SVV/gen3_middleware_acceptance_testing.	
LVV-T1987	1	Pass		
LVV-T1983	1	Pass	For this test execution, we will use the regular monthly (re-)processing of the RC2 dataset to demonstrate the capabilities. The most recent processing was executed with weekly release 'w_2022_12' on the NCSA lsst-devl machines, submitted from path /scratch/brendal4/bps-gen3-rc2/w_2022_12/submit/HSC/runs/RC2/w_2022_12/DM-34125.	
T. Cycle LVV-C162:		**LDM-503-GEN3: Gen 3 Ingest raw dataset**		Not Executed
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T1985	1	Not Executed		
T. Cycle LVV-C190:		**LDM-556: Middleware Acceptance Testing**		In Progress
Test Cases	**Ver.**	**Status**	**Comment**	**Issues**
LVV-T2503	1	Pass		
LVV-T2502	1	Not Executed		
LVV-T2501	1	Not Executed		
LVV-T2499	1	Not Executed		
LVV-T2498	1	Pass		
LVV-T2497	1	Pass		
LVV-T2496	1	Not Executed		
LVV-T2495	1	Pass		
LVV-T2494	1	Pass		
LVV-T2493	1	Pass		
LVV-T2492	1	Pass		

LVV-T2491	1	Pass	Working on lsst-devl machines in a cloned 'daf_butler' repository at /project/jcarlin/SVV/-gen3_middleware_acceptance_testing
LVV-T2488	1	Pass	
LVV-T2487	1	Not Executed	
LVV-T2486	1	Not Executed	
LVV-T2485	1	Not Executed	
LVV-T2484	1	Not Executed	
LVV-T2483	1	Not Executed	
LVV-T2482	1	In Progress	
LVV-T2481	1	Pass	
LVV-T2480	1	Pass	
LVV-T2479	1	Pass	
LVV-T2478	1	Not Executed	
LVV-T2477	1	Not Executed	
LVV-T2476	1	Not Executed	
LVV-T2474	1	Pass	
LVV-T2475	1	Pass	
LVV-T2473	1	Pass	
LVV-T2472	1	Not Executed	
LVV-T2471	1	Pass	
LVV-T2470	1	Pass	We verify this with the same query as used in LVV-T2469, but instead specifying "findFirst=True" to override the default behavior.
LVV-T2469	1	Pass	We verify this by demonstrating that a 'deepCoadd_calexp' can be retrieved for the same tract, patch, band combination, but from different collections (i.e., data processed with different pipeline versions).
LVV-T2468	1	Pass	
LVV-T2466	1	Not Executed	
LVV-T2467	1	Pass	We will verify this by demonstrating that all dataset overlapping a given tract/patch combination (and thus a specific sky region) can be readily discovered.
LVV-T2464	1	Pass	
LVV-T2465	1	Not Executed	
LVV-T2461	1	Not Executed	
LVV-T2463	1	Not Executed	

LVV-T2462	1	Pass	Working on lsst-devl machines in a cloned 'pipe_base' repository at /project/jcarlin/SVV/-gen3_middleware_acceptance_testing/pipe_base
LVV-T2460	1	Pass	Working on lsst-devl machines in a cloned 'pipe_base' repository at /project/jcarlin/SVV/-gen3_middleware_acceptance_testing/pipe_base
LVV-T2457	1	Not Executed	
LVV-T2456	1	Pass	
LVV-T2455	1	Not Executed	
LVV-T2454	1	Not Executed	
LVV-T2458	1	Pass	
LVV-T2451	1	Pass	
LVV-T2453	1	Pass	
LVV-T2449	1	Pass	
LVV-T2452	1	Pass	Working with a cloned 'daf_butler' repository at /project/jcarlin/SVV/-gen3_middleware_acceptance_testing/daf_butler on the lsst-devl machines.
LVV-T2450	1	Not Executed	
LVV-T2447	1	Not Executed	
LVV-T2446	1	Pass	
LVV-T2444	1	Pass	
LVV-T2442	1	Initial Pass	
LVV-T2443	1	Initial Pass	
LVV-T2441	1	Not Executed	
LVV-T2440	1	Pass	
LVV-T2439	1	Pass	

Table 3: 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-C160

Open test cycle *LDM-503-GEN3: Gen 3 Butler Acceptance Testing* in Jira.

Test Cycle name: LDM-503-GEN3: Gen 3 Butler Acceptance Testing
Status: Done

This test cycle is meant to demonstrate that the Gen3 butler and associated database and pipeline interfaces have matured to the point where they can replace the Gen2 butler. The test cases outlined here:

1. use a series of modest pipeline executions to show that the Gen3 software can support all future pipeline development,
2. those pipeline executions also show that a batch processing system (BPS) is available to enable that processing, and
3. demonstrate through inspection that documentation for developers exists
4. confirm that pipeline developers do not know of blockers if all future development assumes Gen3 Butler.

5.1.1 Software Version/Baseline

Not provided.

5.1.2 Configuration

Gen3 Butler repositories with test data are available within DBB spaces. Weekly DM stack has Gen3 and BPS elements present for tests.

5.1.3 Test Cases in LVV-C160 Test Cycle

5.1.3.1 LVV-T2264 - Butler Gen3 maturity sufficient to support future pipeline development.

Version **1**. Open *LVV-T2264* test case in Jira.

This test is meant to verify that Butler Gen3 maturity is sufficient to provide comparable (or better) pipeline capabilities and results to those available under Butler Gen2.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Poll DM developers leads for Data Release and Alert Processing to verify that blockers do not exist if all future development assumed Gen3 Butler's use with Gen2 Butler to be deprecated in the near future.

Expected Result

No known blockers.

Actual Result

Jim Bosch (product owner of DM Middleware) polled both the Science Pipelines team and the Change Control Board to gather feedback. A record of the planning process for deprecation of Gen2 middleware, and comments/feedback on the process, is contained in a Confluence page at `https://confluence.lsstcorp.org/display/DM/Gen2+Middleware+Removal+Planning`, a PDF copy of which is attached to this test execution.

5.1.3.2 LVV-T1984 - Demonstrate documentation/examples of Gen3 usage and cp_pipe equivalent.

Version **1**. Open *LVV-T1984* test case in Jira.

Demonstrate the existence of fundamental documentation necessary to aid Gen2 users with the transition to Gen3 use.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Identify document(s), web-pages, archived presentations, or example notebooks that provide documentation and/or examples of Gen3 functionality.

Test Data

https://pipelines.lsst.io/v/weekly/modules/lsst.cp.pipe/constructing-calibrations.html

Expected Result

Document reference(s) or URL(s) for such documentation.

Actual Result

Usage of the butler and its functionalities is well documented at:
`https://pipelines.lsst.io/modules/lsst.daf.butler/index.html`

Additionally, there is a middleware "frequently asked questions" on the documentation site:
`https://pipelines.lsst.io/middleware/index.html`

A basic data processing tutorial is given in the getting started section of pipelines.lsst.io. In support of Data Preview 0.1, the Community Engagement Team has produced a number of tutorial notebooks demonstrating many

functionalities of the pipelines and middleware, all of which are based on the Gen3 Butler:
`https://github.com/rubin-dp0/tutorial-notebooks`

A detailed guide on how to construct calibrations using cp_pipe is found at:
`https://pipelines.lsst.io/modules/lsst.cp.pipe/constructing-calibrations.html`

5.1.3.3 LVV-T1982 - Run a pipeline on a single node using pipetask.

Version **1**. Open *LVV-T1982* test case in Jira.

To show that individual users have the ability to run either locally (w/ sqlite) or generally (w/ Postgres) using Gen3 Butler infrastructure.

Preconditions:
This test requires that Gen3 Butler infrastructure and underlying pipets have been integrated. It further requires (in spirit) that gen3 schema stability has been reached to facilitate comparison of pipeline results with further stack development can be compared.

Execution status: **Pass**

Final comment:
Working on lsst-devl02, in directory /project/jcarlin/SVV/gen3_middleware_acceptance_testing.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Setup stack, identify inputs, pipetask execution of standard ci_hsc run.

- -

Test Data
ci_hsc raw repository within a Gen3 Butler repo

- -

Expected Result

Pipeline executes standard reduction without failure.

Actual Result

First, set up the science pipelines:

source /software/lsstsw/stack/loadLSST.bash
source scl_source enable devtoolset-8
setup -t w_2022_05 lsst_distrib

Git clone the testdata_ci_hsc and ci_hsc_gen3 repositories (in this case, testdata_ci_hsc is in /project/jcarlin/repos/, and ci_hsc_gen3 is cloned into the working directory for this test campaign).

Checkout the proper tagged version of ci_hsc_gen3 by typing 'git checkout w.2022.05'

Set up the repositories:
setup -j -r /project/jcarlin/repos/testdata_ci_hsc/
setup -j -r /project/jcarlin/SVV/gen3_middleware_acceptance_testing/ci_hsc_gen3/

Now, from the latter of these two directories, execute 'scons', which will run the full end-to-end processing of ci_hsc_gen3:
scons 2>&1 | tee ci_hsc_log_w_2022_05.log

(note that the portions after "scons" serve to pipe the screen output to a log file)

After the execution completed, we examine the output data products and logs to verify that requirements are being met.

LVV-19748: DMS-MWBT-REQ-0020-V-01: Sky Tile Definition

From the output log, we see these lines:

python /software/lsstsw/stack_20220125/stack/miniconda3-py38_4.9.2-1.0.0/Linux64/daf_butler/g1a2eac586c+199d7ac1b3/bin/bu
register-skymap /project/jcarlin/SVV/gen3_middleware_acceptance_testing/ci_hsc_gen3/DATA -C /project/jcarlin/SVV/-

gen3_middleware_acceptance_testing/ci_hsc_gen3/configs/skymap.py
lsst.pipe.tasks.script.registerSkymap INFO: sky map has 1 tracts
lsst.pipe.tasks.script.registerSkymap INFO: tract 0 has corners (321.860, -1.179), (318.875, -1.179), (318.874, 1.806), (321.861, 1.806) (RA, Dec deg) and 16 x 16 patches

This demonstrates how the sky tiling was defined programmatically. Examine the skymap object in the repository:

(lsst-scipipe) [jcarlin@lsst-devl02 ci_hsc_gen3]$ ipython
In [1]: from lsst.daf.butler import Butler
In [2]: butler = Butler('DATA/', collections=['HSC/runs/ci_hsc'])
Load the skymap:
In [3]: skymap = butler.get('skyMap')
In [4]: skymap
Out[4]: <lsst.skymap.discreteSkyMap.DiscreteSkyMap at 0x7f6e0ab3ea60>
Generate a tract using this skymap to confirm that it is well-formed:
In [10]: tract = skymap.generateTract(0)
In [11]: tract
Out[11]: TractInfo(id=0, ctrCoord=[0.770139958995028, -0.6378515275139028, 0.005465565599902794])

We have successfully demonstrated that a skymap (i.e., a "tiling of the sky") can be added programmatically.

LVV-19785: DMS-MWBT-REQ-0046-V-01: External Data Ingest
LVV-19767: DMS-MWBT-REQ-0047-V-01: External Data Ingest and Serve

Confirm that the raw images from the HSC dataset were ingested. First, check the butler registry:

butler query-datasets DATA raw

type run id band instrument detector physical_filter exposure
—- ———– ————————————————– —- ———- ——– ————— ———
raw HSC/raw/all 57e68524-8eff-5787-ad73-9fc0c35fbe41 r HSC 16 HSC-R 903334
raw HSC/raw/all b7ed320f-47ec-5a26-baef-39e2a5fc0dc2 r HSC 22 HSC-R 903334
(output truncated)

The registry knows about the data. Now confirm that the images can be retrieved via the butler:

In [18]: raw = butler.get('raw', {'detector':16, 'exposure':903334})
In [19]: raw
Out[19]: <lsst.afw.image.exposure.ExposureU at 0x7f6df91c8d30>
The image exists as a "ExposureU" object in the repository. Check some of its properties:
In [22]: raw.getDimensions()
Out[22]: Extent2I(2144, 4241)
In [23]: raw.image.array.mean()
Out[23]: 1417.2363656619636
In [24]: raw.image.array.std()
Out[24]: 650.7810120700118

This confirms that the full array of pixel values has been ingested, and that they have (on average) non-zero values.

LVV-19774: DMS-MWST-REQ-0005-V-01: Pipeline configuration

The attached file "DRP.yaml" is the pipeline specification for ci_hsc_gen3 (note that it begins by importing another pipeline). It illustrates the configuration of individual tasks' parameters within the pipeline, thus verifying that this requirement is met.

LVV-19795: DMS-MWST-REQ-0004-V-01: Pipeline specification
LVV-19863: DMS-MWST-REQ-0008-V-01: Use of Tasks and configurations

The attached file "DRP-RC2.yaml" is the pipeline that specifies all of the tasks that are run in ci_hsc_gen3. By examination, it is clear that the pipeline specification satisfies the requirement that it specify the units of code that are to be run, and the order in which they should be executed.

Furthermore, the requirement in DMS-MWST-REQ-0008 that the middleware support organization of work within a step via configurable Tasks is clearly demonstrated by inspection of the attached file "DRP-RC2.yaml", which has been successfully executed as part of this test.

LVV-19864: DMS-MWST-REQ-0017-V-01: Pipeline specification definition

The ability to construct a pipeline specification via configuration is illustrated by the attached file "DRP-RC2.yaml," which fully specifies an end-to-end pipeline for processing of the HSC-RC2 dataset (whose processing has been demonstrated earlier in this test execution).

The requirement that a pipeline specification may be constructed programmatically via Python API is technically met by the 'Pipeline' class in 'pipe_base' (see https://github.com/lsst/pipe_base/blob/main/python/lsst/pipe/base/pipeline.py). In practice, this is rarely used, as it is much more convenient and flexible to construct pipelines via configuration.

LVV-19861: DMS-MWST-REQ-0010-V-01: Executable by supervisory framework

Interpreting this to mean that any pipeline should be executable by the same system, we demonstrate that this requirement is satisfied by our nightly continuous integration (CI) processing jobs that execute in the Jenkins environment. These include 'ci_hsc_gen3' (full end-to-end processing of a small HSC dataset, which has been demonstrated as part of this test execution; see https://ci.lsst.codes/blue/organizations/jenkins/scipipe%2Fci_hsc/activity), and 'ap_verify' (nightly processing of DECam data through the full alert production processing; see https://ci.lsst.codes/blue/organizations/jenkins/scipipe%2Fap_verify/activity). Both are run in the same execution environment, but are completely different and independent pipelines.

LVV-19742: DMS-MWST-REQ-0013-V-01: I/O via Butler

This requirement is satisfied by the design of 'pipelineTask' (see the base class at https://github.com/lsst/pipe_base/blob/main/python/lsst/pipe/base/pipelineTask.py). Examination of the code within that base class demonstrates that the butler I/O is handled by 'pipelineTask'. In particular, the 'runQuantum' method provides a (quoting the docstring from the code) "method to do butler I/O and/or transforms to provide in-memory objects for tasks' run methods."

LVV-19750: DMS-MWST-REQ-0014-V-01: Butler dataset type configuration

An example of specifying input/output dataset types within configuration is seen in the attached pipeline "DRP-ci_hsc+fakes.yaml" (see online at this link). This pipeline is executed nightly as part of the 'ci_hsc' processing. In particular, note the "deblend" task of that pipeline, where the input dataset is specified via 'connections.inputCoaddName: "fakes_deep"', and the output via 'connections.outputCoaddName: "fakes_deep"'. Additional examples can be seen in the 'faro' package, where the preparation_matched_jointcal_fgcm.yaml pipeline specifies output dataset types, which are subsequently specified as inputs to tasks in measurement_matched.yaml.

LVV-19850: DMS-MWST-REQ-0006-V-01: Dataset grouping

The grouping of datasets is typically specified for a given 'pipelineTask' in its Connections class, but these connections are also configurable via pipeline specifications. For example, in the pipeline found in 'drp_pipe/ingredients/DRP.yaml' (see https://github.com/lsst/drp_pipe/blob/main/ingredients/DRP.yaml), the "transformDiaSourceCat" task definition (lines 65-73) specifies the coaddName, diaSourceSchema, diaSourceCat, diffIm, and diaSourceTable. By examination of the source code (https://github.com/lsst/ap_association/blob/78c54e57d4d6669e639377797599f186d046457c/python/lsst/ap/association/transformDiaSourceCatalog.py#L41-L68), one can see that the values in the pipeline are explicit configuration overrides to the connection defaults that are already defined in the Connections class for the ‘transformDiaSourceCatalog‘ task.

LVV-19859: DMS-MWST-REQ-0007-V-01: Changes of parallelization

The attached file “pipeline_detail.png” shows a portion of the directed acyclic graph for the ci_hsc_gen3 pipeline (the full figure is attached as “pipeline.png”). Each step in the processing is illustrated by a box in this figure, with the required data dimensions given in each box. The different groupings of dataset types (and dimensions) in various steps of the processing demonstrates that the Pipeline specification permits each step to have a different required data grouping, as required.

LVV-19796: DMS-MWST-REQ-0011-V-01: Phases of execution

The attached log (“ci_hsc_log_w_2022_05.log”) of the ci_hsc_gen3 processing, as well as the visualization of the processing pipeline in “pipeline.png,” illustrate that the “pre-flight” phase in which a quantum graph (i.e., DAG) is generated is followed by the “run” phase in which the tasks are executed in succession.

LVV-19809: DMS-MWST-REQ-0012-V-01: Implied inputs

To demonstrate that implied inputs can be accessed using fully resolved references retrieved from the quantum graph (DAG), first run ‘pipetask qgraph...‘ to persist the quantum graph.

pipetask qgraph -b “$CI_HSC_GEN3_DIR”/DATA/butler.yaml -p “$DRP_PIPE_DIR/pipelines/HSC/DRP-ci_hsc.yaml” -d “skymap=’discrete/ci_hsc’ AND tract=0 AND patch=69” -i HSC/defaults -o test_qgraph_gen –save-qgraph ci_hsc_gen3_qgraph.qgraph

Now, in an ipython terminal (with the Science Pipelines set up):

In [1]: from lsst.pipe.base import QuantumGraph
In [2]: from lsst.daf.butler import DimensionUniverse

Read the persisted quantum graph:

In [3]: qg = QuantumGraph.loadUri(’ci_hsc_gen3_qgraph.qgraph’, universe=DimensionUniverse())

“Implied inputs” are dataset types that are typically specified as PrerequisiteInputs in connections classes. One place where this occurs is in isrTask, which requires a bias, crosstalk object, dark frame, and other PrerequisiteInputs (i.e., “implied inputs”). For this test, we will show that the bias is retrievable based on datarefs in the DAG we have persisted.

```
In [4]: ctquanta = qg.findQuantaWithDSType('bias')
```

```
In [5]: for q in ctquanta:
...: print(q)
...:
Quantum(taskName=lsst.ip.isr.isrTask.IsrTask, dataId={instrument: 'HSC', detector: 1, exposure: 904014, ...})
Quantum(taskName=lsst.ip.isr.isrTask.IsrTask, dataId={instrument: 'HSC', detector: 18, exposure: 903338, ...})
Quantum(taskName=lsst.ip.isr.isrTask.IsrTask, dataId={instrument: 'HSC', detector: 1, exposure: 903346, ...})
Quantum(taskName=lsst.ip.isr.isrTask.IsrTask, dataId={instrument: 'HSC', detector: 0, exposure: 903344, ...})
Quantum(taskName=lsst.ip.isr.isrTask.IsrTask, dataId={instrument: 'HSC', detector: 4, exposure: 904010, ...})
...
```

This demonstrates that the quanta using a "bias" dataset type can be accessed. A similar query can be done to see the tasks that use the biases:

```
In [6]: cttasks = qg.tasksWithDSType('bias')
```

```
In [7]: for t in cttasks:
...: print(t)
...:
TaskDef(lsst.ip.isr.isrTask.IsrTask, label=isr)
```

As expected, IsrTask uses the biases. Now extract one of these quanta for further exploration:

```
In [8]: q0 = ctquanta.pop()
```

```
In [9]: q0.dataId
Out[9]: {instrument: 'HSC', detector: 1, exposure: 904014, ...}
```

Instantiate a butler pointing to the ci_hsc_gen3 repository:

```
In [10]: from lsst.daf.butler import Butler
In [11]: butler = Butler('DATA/butler.yaml', collections=['HSC/runs/ci_hsc'])
```

Now, for the quantum of interest, we can get the 'DatasetRef' of the input bias, then load the bias from the butler using this dataset reference:

In [12]: (ref,) = q0.inputs["bias"]
In [13]: bias = butler.getDirect(ref)

In [14]: bias
Out[14]: <lsst.afw.image.exposure.ExposureF at 0x7f6f3c12be70>

This confirms that the bias we have retrieved is an "ExposureF" type object. Confirm that it has non-zero values and an associated bounding box:

In [15]: bias.image
Out[15]:
lsst.afw.image.image.ImageF=[[2.7007866 -0.14668204 0.69989073 ... 0.71999675 0.25799465
2.7209432]
[2.3166523 0.3923351 -0.19359061 ... 0.79756474 -0.5102319
2.9523087]
[3.4711118 -0.63503593 0.31618226 ... -0.04773628 -0.12536396
3.0303118]
...
[2.9787798 -0.2536826 0.13162205 ... 0.87448245 -0.5876055
4.1830816]
[2.7134619 -0.40717614 0.15982895 ... 0.9519046 0.38719296
3.375857]
[3.67328 0.5943649 0.5933383 ... 1.5677603 0.33683997
2.7228777]], bbox=(minimum=(0, 0), maximum=(2047, 4175))

In [16]: bias.getDimensions()
Out[16]: Extent2I(2048, 4176)

We have thus demonstrated that the DAG contains fully resolved references to implied inputs related to the 'Isr-Task'.

5.1.3.4 LVV-T1987 - Run Calibration Products Processing (CPP)

Version **1**. Open *LVV-T1987* test case in Jira.

Demonstrate that basic calibration processing from Gen2 era has been enabled within Gen3 environment. This test is not concerned with large scales but merely demonstrates that Gen3 capability to generate calibration products (i.e. they are no longer required to be generated in Gen2 and then migrated to Gen3).

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Identify an existing or instantiate a new Gen3 repository with raw bias, dark, and flat observations.

Test Data

It is preferred these data be early observatory products (i.e. either AuxTel/LATISS or ComCam).

Expected Result

A Gen3 repo with appropriate raw data products.

Actual Result

We use calibrations from LATISS that are stored in the shared repository at NCSA.

First, we identify a set of exposures to use as inputs from the repository:

'butler query-dimension-records /repo/main exposure –where "instrument='LATISS' AND exposure.observation_type='bias' AND exposure.target_name='Park position' AND exposure.exposure_time=0.0 AND exposure.dark_time < 0.1 AND exposure.day_obs > 20210101"'

We will use "rerun" 20210702a, and select a subset from the list of all bias exposures returned by the command above: 2021012000020, 2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010

List of dark exposures: 2021021700078, 2021021700080, 2021021800057, 2021030900054, 2021030900060
List of flats (with "RG610" filter): 2021052500077, 2021052500078, 2021052500079, 2021052500080, 2021052500081

Step 2 Step Execution Status: **Pass**

Description

Create master bias, dark and flat products from the raw products.

Expected Result

A master bias, dark, and flat calibration product.

Actual Result

For this test, we follow examples documented at `https://pipelines.lsst.io/v/daily/modules/lsst.cp.pipe/constructing-calibrat html`. To confirm that each combined calibration frame is well-formed, we use the 'cp_verify' package, which determines the quality of newly produced calibrations in an automated way rather than visual inspection.

Create the master bias:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_PIPE_DIR/pipelines/Latiss/cpBias.yaml -i LATISS/raw/all,LATISS/calib -o u/jcarlin/biasGen.20210702a -d "instrument='LATISS' AND detector=0 AND exposure IN (2021012000020, 2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010)" -c isr:doDefect=False -c isr:doEmpiricalReadNoise=True 2>&1 | tee bias.20210702a.log

Verify the bias:

pipetask run -b /repo/main/butler.yaml -p $CP_VERIFY_DIR/pipelines/VerifyBias.yaml
-i u/jcarlin/biasGen.20210702a,LATISS/raw/all,LATISS/calib -o u/jcarlin/verifyBias.20210702a
-d "instrument='LATISS' AND detector=0 AND
exposure IN (2021012000020, 2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010)"

Certify the bias:

Usage: butler certify-calibrations [OPTIONS] REPO INPUT_COLLECTION
OUTPUT_COLLECTION DATASET_TYPE_NAME
We'll use INPUT_COLLECTION=u/jcarlin/biasGen.20210702a and OUTPUT_COLLECTION=u/jcarlin/LATISS/calib

butler certify-calibrations /repo/main/butler.yaml u/jcarlin/biasGen.20210702a u/jcarlin/LATISS/calib –begin-date
2020-01-01 –end-d\
ate 2050-01-01 bias

Construct the defects mask:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_PIPE_DIR/pipelines/Latiss/findDefects.yaml -i LATISS/raw/all,u/jcarlin/biasG
-o u/jcarlin/defectGen.20210706h -d "instrument='LATISS' AND detector=0 AND exposure IN (2021012000020,
2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010)" 2>&1 | tee defect.20210706h.log

Verify the defects:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_VERIFY_DIR/pipelines/VerifyDefect.yaml -i LATISS/raw/all,u/jcarlin/defectGe
.20210706h,u/jcarlin/biasGen.20210702a,u/jcarlin/LATISS/calib -o u/jcarlin/verifyDefect.20210706h -d "instrument='LATISS'
AND detecto\
r=0 AND exposure IN (2021012000020, 2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010)"
2>&1 | tee defectVerify.20210706h.log

Rerun bias verification with the new defects:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_VERIFY_DIR/pipelines/VerifyBias.yaml \
-i LATISS/raw/all,u/jcarlin/defectGen.20210706h,u/jcarlin/biasGen.20210702a,u/jcarlin/LATISS/calib \
-o u/jcarlin/verifyBias.20210706h \
-d "instrument='LATISS' AND detector=0 AND
exposure IN (2021012000020, 2021012000032, 2021012000055, 2021012000061, 2021012100060, 2021031100010)"
\
-c verifyBiasApply:doDefect=True 2>&1 | tee biasVerify.20210706h.log

Certify the new bias for use:

butler certify-calibrations /repo/main/butler.yaml u/jcarlin/defectGen.20210706h u/jcarlin/LATISS/calib \
–begin-date 2020-01-01 –end-date 2050-01-01 defects

Construct the dark frame:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_PIPE_DIR/pipelines/cpDark.yaml \
-i LATISS/raw/all,u/jcarlin/defectGen.20210706h,u/jcarlin/biasGen.20210702a,u/jcarlin/LATISS/calib \
-o u/jcarlin/darkGen.20210707a
-d "instrument='LATISS' AND detector=0 AND
exposure IN (2021021700078, 2021021700080, 2021021800057, 2021030900054, 2021030900060)" \
-c isr:doCrosstalk=False 2>&1 | tee dark.20210707a.log

Verify the dark:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_VERIFY_DIR/pipelines/VerifyDark.yaml \
-i LATISS/raw/all,u/jcarlin/darkGen.20210707a,u/jcarlin/defectGen.20210706h,u/jcarlin/biasGen.20210702a,u/jcarlin/LATISS/calib \
-o u/jcarlin/verifyDark.20210707a -d "instrument='LATISS' AND detector=0 AND
exposure IN (2021021700078, 2021021700080, 2021021800057, 2021030900054, 2021030900060)" \
-j 4 2>&1 | tee ./darkVerify.20170707a.log

Certify the dark:

butler certify-calibrations /repo/main/butler.yaml u/jcarlin/darkGen.20210707a u/jcarlin/LATISS/calib \
–begin-date 2020-01-01 –end-date 2050-01-01 dark

Construct the flat frame:
pipetask –long-log run -b /repo/main/butler.yaml -p $CP_PIPE_DIR/pipelines/Latiss/cpFlat.yaml -i LATISS/raw/all,u/jcarlin/defectGen.20210706h,u/jcarlin/biasGen.20210702a,u/jcarlin/darkGen.20210707a,u/jcarlin/LATISS/calib -o u/jcarlin/flatGen.20210707a
-d "instrumen\
t='LATISS' AND detector=0 AND exposure IN (2021052500077, 2021052500078, 2021052500079, 2021052500080, 2021052500081)" -j 4 2>&1 | tee flat.20210707a.log

Verify the flat:

pipetask –long-log run -b /repo/main/butler.yaml -p $CP_VERIFY_DIR/pipelines/VerifyFlat.yaml -i LATISS/raw/all,u/jcarlin/defectGen.2 -o u/jcarlin/verifyFlat.20210707a -d "instrument='LATISS' AND detector=0 AND exposure in (2021052500077, 2021052500078, 2021052500079, 2021052500080, 2021052500081)" -j 4 2>&1 | tee ./flatVerify.20210707a.log

Certify the flat:

butler certify-calibrations /repo/main/butler.yaml u/jcarlin/flatGen.20210707a u/jcarlin/LATISS/calib –begin-date 2020-01-01 –end-date 2050-01-01 flat

We have now successfully processed raw calibration frames to produce a master bias, dark, and flat calibration product, and verified that these are well-formed and suitable for use.

5.1.3.5 LVV-T1983 - Mini RC2 processing capability

Version **1**. Open *LVV-T1983* test case in Jira.

Demonstrate that a typical 3-tract RC2 data processing is possible using the Gen3 system and the nascent Batch Production Service (BPS). This test is meant to demonstrate that Gen3 + BPS systems are capable of supporting future DM development by demonstrating that processing routinely used by developers for benchmarking/testing improvements can be performed in a reasonable time.

Preconditions:

Execution status: **Pass**

Final comment:
For this test execution, we will use the regular monthly (re-)processing of the RC2 dataset to demonstrate the capabilities. The most recent processing was executed with weekly release 'w_2022_12' on the NCSA lsst-devl machines, submitted from path /scratch/brendal4/bps-gen3-rc2/w_2022_12/submit/HSC/runs/RC2/w_2022_12/DM-34125.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

setup environment
identify input data products

Test Data

RC2 raw repo

Example Code

for example on lsstdev-* resources at NCSA

export lsstsw_root=/software/lsstsw/stack
export EUPS_TAG="w_2020_46"

source /opt/rh/devtoolset-8/enable
source ${lsstsw_root}/loadLSST.bash
setup lsst_distrib -t ${EUPSTAG}

Expected Result

software environment ready for job submission

Actual Result

A setup script containing the following lines was first executed:

EUPSTAG=w_2022_12
lsstsw_root=/software/lsstsw/stack
#lsstsw_root=/software/lsstsw/stack_20210520

echo ${EUPSTAG}
source /opt/rh/devtoolset-8/enable
source ${lsstsw_root}/loadLSST.bash
setup lsst_distrib -t ${EUPSTAG}

export OMP_NUM_THREADS=1

auth_path=/home/brendal4/.lsstauth

Postgres
export LSST_DB_AUTH=$auth_path/db-auth-rc.yaml
#export DAF_BUTLER_CONFIG_PATH='pwd'
export PGPASSFILE=$auth_path/.pgpass

HTCondor API
export PYTHONPATH=$PYTHONPATH:/usr/lib64/python3.6/site-packages

Now, we confirm that the correct weekly pipeline version is set up:

$ eups list -s lsst_distrib
g64fc59b30a+c0e88c3db7 w_2022_12 setup

This demonstrates that the pipelines from w_2022_12 were properly set up.

Step 2 Step Execution Status: **Pass**

Description
BPS pipeline submission

Example Code
pipetask qgraph -d "tract = 9615 and instrument='HSC' and skymap='hsc_rings_v1'" \
-b {gen3_repo}/{version}/butler.yaml \
-i HSC/calib,HSC/raw/all,HSC/masks,refcats,skymaps \
-p /home/madamow/gen2-to-gen3/bps/HSC-RC2.yaml \
-q /home/madamow/gen2-to-gen3/bps/submit/RC2/w_2020_42/DM-27244/20201102T10h22m03s/RC2_w_2020_42_DM-27244_20201102T10h22m03s.pickle

---

Expected Result

Pipeline execution is successful. An estimate of the compute resources used (# cores, memory, wall time) for each execution should be reported.

Actual Result

The processing is executed in a series of “steps” – an example script for “step 2” of the pipeline looks like:

##############
#BPS OPERATOR
##############

pipelineYaml: “$OBS_SUBARU_DIR/pipelines/DRP.yaml#step2”

bpsUseShared: True

campaign: “G3W12”
computeSite: ncsapool
requestMemory: 2048
requestCpus: 1

payload:
payloadName: rc2
butlerConfig: /repo/main/butler.yaml
inCollection: HSC/RC2/defaults
output: “HSC/runs/RC2/w_2022_12/DM-34125”
dataQuery: “”
dataQuery: “instrument=’HSC’ and skymap=’hsc_rings_v1’ and detector!=9 and detector.purpose=’SCIENCE’”

#createQuantumGraph: pipetask qgraph -d “{dataQuery}” -b {butlerConfig} -i {inCollection} –output {output} –output-run {outCollection} -p {pipelineYaml} -q {qgraphFile} –qgraph-dot {qgraphFile}.dot

extraRunQuantumOptions: –no-versions
extraInitOptions: –no-versions

pipetask:
assembleCoadd:
requestMemory: 8192
makeWarp:
requestMemory: 85000
jointcal:
requestMemory: 21000
skyCorr:
requestMemory: 15500
deblend:
requestMemory: 15500

After executing all steps, the following table shows (a) that all tasks completed successfully, and (b) summary statistics of the number of quanta for each task, the runtime, and maximum memory request.

	nQuanta	startTime	cpu sec/job	cpu-hours	MaxRSS GB
20220319T213338Z_isr	44496	2022-03-20 02:16:10	48.81	25 days, 3:15:51.758926	1798.99
20220319T213338Z_characterizeImage	44496	2022-03-20 05:54:54	76.77	39 days, 12:52:43.901541	823.62
20220319T213338Z_calibrate	44496	2022-03-20 11:39:20	43.85	22 days, 13:58:56.606125	845.43
20220319T213338Z_writeSourceTable	44496	2022-03-20 14:52:58	12.07	6 days, 5:08:40.185788	580.18
20220319T213338Z_transformSourceTable	44496	2022-03-20 17:07:37	11.9	6 days, 3:07:33.742485	580.19
20220321T153517Z_consolidateVisitSummary	432	2022-03-21 17:26:31	53.8	6:27:20.826998	555.1
20220321T153517Z_consolidateSourceTable	432	2022-03-21 17:29:58	19.65	2:21:30.447276	1391.13
20220321T153517Z_TE3	432	2022-03-21 17:38:00	48.12	5:46:28.574781	3693.29
20220321T153517Z_TE4	432	2022-03-21 17:41:27	49.03	5:53:02.669183	3735.93
20220321T153517Z_nsrcMeasVisit	432	2022-03-21 17:42:16	44.89	5:23:12.090092	3500.46
20220321T153517Z_fgcmBuildStarsTable	1	2022-03-21 18:15:30	1423.03	0:23:43.033440	5562.68
20220321T153517Z_fgcmFitCycle	1	2022-03-21 18:34:29	2533.93	0:42:13.934329	14950.0
20220321T153517Z_skyCorr	432	2022-03-21 19:16:28	1036.5	5 days, 4:22:46.138239	11938.41
20220321T153517Z_fgcmOutputProducts	1	2022-03-21 19:39:03	54.41	0:00:54.413377	4143.73
20220321T222013Z_selectGoodSeeingVisits	1281	2022-03-22 00:01:58	9.64	3:25:43.145125	316.54
20220321T222013Z_jointcal	16	2022-03-22 00:03:11	1073.57	4:46:17.194291	7271.61
20220321T222013Z_matchCatalogsPatch	1203	2022-03-22 00:25:08	30.07	10:02:51.051603	4824.86
20220321T222013Z_makeWarp	41700	2022-03-22 00:51:21	72.36	34 days, 22:07:41.057967	2221.89
20220321T222013Z_matchCatalogsTract	15	2022-03-22 01:21:01	524.12	2:11:01.730450	56072.13
20220321T222013Z_matchCatalogsPatchMultiBand	163	2022-03-22 01:31:23	81.25	3:40:43.409102	4774.41
20220321T222013Z_assembleCoadd	1281	2022-03-22 01:52:01	553.23	8 days, 4:51:33.264997	4103.96
20220321T222013Z_AM2	15	2022-03-22 02:13:42	314.71	1:18:40.660123	803.74
20220321T222013Z_AM3	15	2022-03-22 02:13:46	26.51	0:06:37.700437	766.76
20220321T222013Z_PF1	15	2022-03-22 02:18:23	16.29	0:04:04.280209	524.59
20220321T222013Z_PA1	15	2022-03-22 02:20:05	16.27	0:04:04.078226	524.11
20220321T222013Z_AF2	15	2022-03-22 02:21:14	345.4	1:26:21.011795	1768.62
20220321T222013Z_psfPhotRepStar3	15	2022-03-22 02:25:11	30.38	0:07:35.758447	1461.2
20220321T222013Z_AF3	15	2022-03-22 02:26:22	26.6	0:06:38.973148	760.79
20220321T222013Z_modelPhotRepStar4	15	2022-03-22 02:26:31	29.67	0:07:25.065579	1463.18
20220321T222013Z_modelPhotRepStar3	15	2022-03-22 02:26:34	30.06	0:07:30.949140	1459.19
20220321T222013Z_modelPhotRepGal4	15	2022-03-22 02:33:45	86.8	0:21:41.951140	6595.79
20220321T222013Z_AF1	15	2022-03-22 02:55:01	135.09	0:33:46.370491	1066.98
20220321T222013Z_AD1	15	2022-03-22 02:55:15	135.48	0:33:52.247320	1075.14
20220321T222013Z_modelPhotRepStar1	15	2022-03-22 02:56:18	39.44	0:09:51.597720	1467.83
20220321T222013Z_modelPhotRepGal1	15	2022-03-22 03:03:12	116.4	0:29:06.019895	6617.82
20220321T222013Z_templateGen	1281	2022-03-22 03:06:30	275.58	4 days, 2:03:36.889334	3962.54
20220321T222013Z_psfPhotRepStar2	15	2022-03-22 03:10:19	34.77	0:08:41.517583	1457.18
20220321T222013Z_AB1	814	2022-03-22 03:11:20	13.13	2:58:09.316412	356.88
20220321T222013Z_modelPhotRepStar2	15	2022-03-22 03:11:40	33.91	0:08:28.629201	1463.16
20220321T222013Z_modelPhotRepGal3	15	2022-03-22 03:16:53	93.53	0:23:22.956956	6595.7
20220321T222013Z_AD3	15	2022-03-22 03:21:08	26.17	0:06:32.515903	760.81
20220321T222013Z_modelPhotRepGal2	15	2022-03-22 03:58:39	111.79	0:27:56.776343	6593.73
20220321T222013Z_healSparsePropertyMaps	16	2022-03-22 04:09:00	1381.54	6:08:24.637536	1457.97
20220321T222013Z_psfPhotRepStar1	15	2022-03-22 04:11:57	38.99	0:09:44.919765	1474.62
20220321T222013Z_detection	1281	2022-03-22 04:18:27	84.74	1 day, 6:09:12.529867	1502.65
20220321T222013Z_psfPhotRepStar4	15	2022-03-22 04:45:23	29.91	0:07:28.585883	1463.16
20220321T222013Z_AM1	15	2022-03-22 04:55:05	124.49	0:31:07.369815	767.7
20220321T222013Z_deblend	241	2022-03-22 05:01:05	3190.23	8 days, 21:34:05.114494	14171.61
20220321T222013Z_mergeDetections	241	2022-03-22 05:06:55	91.81	6:08:46.634552	467.64
20220321T222013Z_measure	1281	2022-03-22 11:25:21	4173.24	61 days, 20:58:39.947048	4924.52
20220321T222013Z_mergeMeasurements	241	2022-03-22 12:17:21	30.63	2:03:00.796036	4244.76
20220321T222013Z_transformObjectTable	241	2022-03-22 16:19:11	43.31	2:53:58.320545	1231.31
20220321T222013Z_forcedPhotCoadd	1281	2022-03-22 17:31:57	7452.97	110 days, 12:00:56.815577	3038.76
20220321T222013Z_writeObjectTable	241	2022-03-22 20:20:46	78.62	5:15:47.556766	14817.58
20220321T222013Z_AD2	15	2022-03-23 14:24:37	344.77	1:26:11.611876	1732.58
20220323T173939Z_getTemplate	29361	2022-03-24 00:06:42	28.48	9 days, 16:16:06.199511	1795.09
20220323T173939Z_forcedPhotCcd	57073	2022-03-24 02:12:35	200.77	132 days, 14:58:03.819236	1811.78
20220323T173939Z_imageDifference	29324	2022-03-24 04:54:13	71.16	24 days, 3:37:28.295211	3767.84
20220323T173939Z_transformDiaSourceCat	29324	2022-03-24 10:39:51	11.12	3 days, 18:34:53.116333	492.05
20220323T173939Z_forcedPhotDiffim	57010	2022-03-24 11:51:40	204.91	135 days, 5:00:19.708768	1777.93
20220323T173939Z_writeForcedSourceTable	57010	2022-03-24 14:13:28	11.66	7 days, 16:41:54.091953	594.12
20220324T205113Z_transformForcedSourceTable	240	2022-03-24 21:51:45	63.14	4:12:32.824279	1206.38
20220324T205113Z_drpAssociation	240	2022-03-24 21:52:00	218.38	14:33:31.986746	843.65
20220324T205113Z_TE1	14	2022-03-24 21:54:04	335.3	1:18:14.155728	38451.4
20220324T205113Z_TE2	16	2022-03-24 21:55:14	333.89	1:29:02.180038	38433.41
20220324T205113Z_consolidateForcedSourceTable	2	2022-03-24 22:01:22	139.09	0:04:38.181782	39206.5
20220324T205113Z_forcedPhotDiffOnDiaObjects	55731	2022-03-24 22:33:38	24.49	15 days, 19:10:43.122144	672.01
20220324T205113Z_forcedPhotCcdOnDiaObjects	55731	2022-03-24 22:34:07	38.21	24 days, 15:30:40.270414	673.6
20220324T205113Z_drpDiaCalculation	240	2022-03-24 22:51:30	370.56	1 day, 0:42:13.642205	730.45
20220324T205113Z_consolidateFullDiaObjectTable	3	2022-03-24 22:57:02	24.36	0:01:13.084340	9909.66
20220324T205113Z_consolidateAssocDiaSourceTable	3	2022-03-24 23:13:24	25.02	0:01:15.061400	8329.09
20220324T205113Z_transformForcedSourceOnDiaObjectTable	240	2022-03-25 00:42:35	39.54	2:38:09.917989	1278.93
20220324T205113Z_writeForcedSourceOnDiaObjectTable	55731	2022-03-25 01:27:50	10.53	6 days, 18:56:39.681912	673.1
20220324T205113Z_consolidateForcedSourceOnDiaObjectTable	2	2022-03-25 01:47:51	85.56	0:02:51.119232	35856.87
20220324T205113Z_wPerp	2	2022-03-25 14:55:27	1370.25	0:45:40.509903	100791.68
20220325T211319Z_consolidateDiaSourceTable	432	2022-03-25 21:20:30	8.77	1:03:07.355830	400.48
20220325T213046Z_consolidateHealSparsePropertyMaps	6	2022-03-25 21:32:03	21.44	0:02:08.610631	669.23
20220325T213046Z_makeVisitTable	1	2022-03-25 21:32:05	46.8	0:00:46.799852	265.98
20220325T213046Z_makeCcdVisitTable	1	2022-03-25 21:32:09	49.63	0:00:49.626864	328.31
20220321T222013Z_consolidateObjectTable	3	2022-04-05 18:51:16	100.81	0:05:02.420258	19462.99
20220321T222013Z_matchCatalogsTractGxsSNR5to80	15	2022-04-08 00:40:29	0.0	0:00:00	0.0
20220321T222013Z_matchCatalogsTractStarsSNR5to80	15	2022-04-08 00:40:29	0.0	0:00:00	0.0
20220321T222013Z_matchCatalogsTractMag17to21p5	15	2022-04-08 00:40:29	0.0	0:00:00	0.0
20220321T222013Z_isolatedStarAssociation	3	2022-04-08 00:40:29	0.0	0:00:00	0.0
campaign	706026	2022-04-08 00:41:13	0.0	700 days, 14:00:25.063836	100791.68

5.2 Test Cycle LVV-C162

Open test cycle *LDM-503-GEN3: Gen 3 Ingest raw dataset* in Jira.

Test Cycle name: LDM-503-GEN3: Gen 3 Ingest raw dataset
Status: Not Executed

In the context of the milestone LDM-503-GEN3, Gen 3 Butler readiness, this test cycle is defining the configuration and the dataset for running a generic **Raw Data Ingestion Into Gen3 Butler** test case. There are currently 5 data sources that require verification as they are the central products that will be produced by Rubin or are used as precursor sets in the development/verification of the data management software systems. The current raw data products that are deemed central to DM development and testing are those from AuxTel/LATISS, ComCam, and precursor data from HyperSuprimeCam (HSC). Note, further tests using LSSTCam (currently only preliminary BOT data from the SLAC test stand are available) or precursor sets from the Dark Energy Camera (DECam) could be added but since these types do not exactly fit the central model used for LSST they are not tied directly to requirements.

5.2.1 Software Version/Baseline

LSST DM Stack with Gen3 Butler.

5.2.2 Configuration

Three separate raw data types, those from: AuxTel/LATISS, ComCam, and HSC (e.g. a CI_HSC raw) should be ingested when this test is executed.

5.2.3 Test Cases in LVV-C162 Test Cycle

5.2.3.1 LVV-T1985 - Verify daf_butler raw data ingest

Version **1**. Open *LVV-T1985* test case in Jira.

Demonstrate that a raw data type can be successfully ingested into a Butler repository.

Preconditions:
In order to run this test, a Gen3 daf butler should be deployed and ready to use, with access to the filesystems where the raw data to ingest is stored.

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Identify data for ingestion HSC RC2 and make a copy at a location for the test. While a suggestion is provided in /project/shared/hsc/COSMOS/2014-03-27/ for a location where such data can be found, the actual data used can be left to the discretion of the person(s) executing the test with the added suggestion that relatively recent data are more likely to reflect the current observatory system state.

Test Data

/project/shared/hsc/COSMOS/2014-03-27/

Expected Result

One or more raw data sets are identified and made available.

Actual Result

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

Description

Identify data for ingestion AuxTel/LATISS and make a copy at a location for the test. While a suggestion is provided in /project/shared/auxTel/_parent/raw/2021-03-23/ for a location where such data can be found, the actual data used can be left to the discretion of the person(s) executing the test with the added suggestion that relatively recent data are more likely to reflect the current observatory system state.

Test Data

/project/shared/auxTel/_parent/raw/2021-03-23/

Expected Result
One or more raw data sets are identified and made available.

Actual Result

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

Description
Identify data for ingestion ComCam and make a copy at a location for the test. While a suggestion is provided in /project/shared/comCam/_parent/raw/2021-05-14/2021051400003/ for a location where such data can be found, the actual data used can be left to the discretion of the person(s) executing the test with the added suggestion that relatively recent data are more likely to reflect the current observatory system state.

Test Data
/project/shared/comCam/_parent/raw/2021-05-14/2021051400003/

Expected Result
One or more raw data sets are identified and made available.

Actual Result

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

Description
Verify that Butler repository is available for the HSC RC2 (Note this needs to be a test repository rather than the central repository as the raw data should not already be present in the repository for this test.)

Test Data
url 1

Example Code

create empty Gen3 repo (for ComCam data)

butler create repo
butler register-instrument repo lsst.obs.lsst.LsstComCam

Expected Result

Actual Result

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

Description

Verify that Butler repository is available for the AuxTel/LATISS (Note this needs to be a test repository rather than the central repository as the raw data should not already be present in the repository for this test.)

Test Data

url 2

Example Code

create empty Gen3 repo (for ComCam data)

butler create repo
butler register-instrument repo lsst.obs.lsst.LsstComCam

Expected Result

Actual Result

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

Description

Verify that Butler repository is available for the ComCam (Note this needs to be a test repository rather than the central repository as the raw data should not already be present in the repository for this test.)

Test Data
url 3

Example Code
create empty Gen3 repo (for ComCam data)

butler create repo
butler register-instrument repo lsst.obs.lsst.LsstComCam

Expected Result

Actual Result

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

Description
Ingest HSC RC2 raw data into repo

Test Data
url 1

Example Code
butler ingest-raws repo raw

Expected Result
Tool reports data ingest successful for HSC RC2 into url 1

Actual Result

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

Description

Ingest AuxTel/LATISS raw data into repo

Test Data

url 2

Example Code

butler ingest-raws repo raw

Expected Result

Tool reports data ingest successful for AuxTel/LATISS into url 2

Actual Result

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

Description

Ingest ComCam raw data into repo

Test Data

url 3

Example Code

butler ingest-raws repo raw

Expected Result

Tool reports data ingest successful for ComCam into url 3

Actual Result

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

Description
Query repository to verify that ingestion of HSC RC2 occurred.

Test Data
url 1

Expected Result
HSC RC2 data are found by query.

Actual Result

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

Description
Query repository to verify that ingestion of AuxTel/LATISS occurred.

Test Data
url 2

Expected Result
AuxTel/LATISS data are found by query.

Actual Result

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

Description
Query repository to verify that ingestion of ComCam occurred.

Test Data
url 3

Expected Result

ComCam data are found by query.

Actual Result

5.3 Test Cycle LVV-C190

Open test cycle *LDM-556: Middleware Acceptance Testing* in Jira.

Test Cycle name: LDM-556: Middleware Acceptance Testing
Status: In Progress

5.3.1 Software Version/Baseline

Not provided.

5.3.2 Configuration

Not provided.

5.3.3 Test Cases in LVV-C190 Test Cycle

5.3.3.1 LVV-T2503 - Verify Outputs from Test Processing Runs

Version **1**. Open *LVV-T2503* test case in Jira.

Verify that the Data Output System interface is usable by algorithmic code being run for test/development purposes, on both development compute environments at the archive center and in personal environments.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Demonstrated by regular reprocessing runs at NCSA.

Expected Result

Actual Result

This capability is demonstrated by the regular monthly (re-)processing of the RC2 dataset. The most recent processing was executed with weekly release ‘w_2022_12‘ on the NCSA lsst-devl machines; details can be found in Jira ticket DM-34125.

Data quality metrics measured on these data appear on April 14 in the following screenshot from our metric monitoring dashboard:

5.3.3.2 LVV-T2502 - Verify Outputs from Science Platform

Version **1**. Open *LVV-T2502* test case in Jira.

Verify that the Data Output System interface shall be usable by algorithmic code run in the Science Platform

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description
Demonstrated by any Science Platform notebook that uses the butler.

Expected Result

Actual Result

5.3.3.3 LVV-T2501 - Verify Outputs from Data Release Production

Version **1**. Open *LVV-T2501* test case in Jira.

Verify that the Data Output System interface is usable by algorithmic code being run as part of Data Release Production.

Demonstrated by regular reprocessing runs at NCSA and DP0.2 production.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.4 LVV-T2499 - Verify Consistent Output Interface

Version **1**. Open *LVV-T2499* test case in Jira.

Verify that the Data Output System provides a consistent interface for writing InMemoryDatasets to storage given a DatasetRef across different types of DataRepositories.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.5 LVV-T2498 - Verify Writing FITS tables

Version **1**. Open *LVV-T2498* test case in Jira.

Verify that the Data Output System is able to write in-memory table objects as FITS files.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute unit tests “test_fitsTables.cc” and “test_fits.py” in lsst.afw package.

Expected Result

Unit tests pass.

Actual Result

Executed ‘scons‘ in a cloned version of the ‘afw‘ package on lsst-devl at NCSA. Here is the relevant line from the log indicating that the C++ unit tests in “test_fitsTables.cc” passed:

running tests/test_fitsTables... passed

Next, execute the ‘test_fits.py‘ unit test on its own:

pytest -s -vv –no-header –cache-clear tests/test_fits.py

Results:

tests/test_fits.py::FLAKE8 PASSED
tests/test_fits.py::FitsTestCase::testIgnoreKeywords PASSED
tests/test_fits.py::FitsTestCase::testReadBlankKeywordComment PASSED
tests/test_fits.py::FitsTestCase::testReadUndefined lsst.afw.fits WARN: In void lsst::afw::fits::{anonymous}::MetadataIterationFuncto
string&, const string&), dropping undefined value for key ‘ADC-STR’.
lsst.afw.fits WARN: In void lsst::afw::fits::{anonymous}::MetadataIterationFunctor::add(const string&, T, const string&) [with T = double; std::string = std::__cxx11::basic_string<char>], replacing undefined value for key ‘DOM-WND’.
PASSED
tests/test_fits.py::FitsTestCase::testSimpleIO PASSED
tests/test_fits.py::FitsTestCase::testUndefinedVector PASSED
tests/test_fits.py::TestMemory::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/miniconda3-py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py PASSED

Thus the writing of FITS tables has been demonstrated.

5.3.3.6 LVV-T2497 - Verify Writing FITS images

Version **1**. Open *LVV-T2497* test case in Jira.

Verify that the Data Output System is able to write in-memory image objects as FITS files

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute unit tests "test_imageIo1.py" and "test_imageIo2.py" in lsst.afw package.

Expected Result

Unit tests pass.

Actual Result

First, clone and set up the 'afwdata' package:
git clone https://github.com/lsst/afwdata.git
cd afwdata
setup -j -r .

Now navigate to a cloned, set up repository of 'lsst.afw' on the lsst-devl machines at NCSA, and execute the following:

pytest -s -vv –no-header –cache-clear tests/test_imageIo1.py

Results:
tests/test_imageIo1.py::FLAKE8 PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testBBoxFromMetadata PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testF32 PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testF64 PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testImageCompressionDisabled PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testLongStrings PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testMEF PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testReadFitsWithOptions PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testS16 PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testSubimage PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testU16 PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testWriteBool PASSED
tests/test_imageIo1.py::ReadFitsTestCase::testWriteReadF64 PASSED
tests/test_imageIo1.py::TestMemory::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/miniconda3-py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py PASSED

pytest -s -vv –no-header –cache-clear tests/test_imageIo2.py

Results:
tests/test_imageIo2.py::FLAKE8 PASSED
tests/test_imageIo2.py::ImageIoTestCase::testFloatCompressedLossless PASSED
tests/test_imageIo2.py::ImageIoTestCase::testFloatCompressedManual SKIPPED (Fix deferred to DM-15644)
tests/test_imageIo2.py::ImageIoTestCase::testFloatCompressedRange SKIPPED (Fix deferred to DM-15644)
tests/test_imageIo2.py::ImageIoTestCase::testFloatUncompressed PASSED
tests/test_imageIo2.py::ImageIoTestCase::testIntegerCompression PASSED
tests/test_imageIo2.py::ImageIoTestCase::testIntegerUncompression PASSED
tests/test_imageIo2.py::ImageIoTestCase::testUInt64 PASSED
tests/test_imageIo2.py::TestMemory::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/miniconda3-py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py P

We have now demonstrated the reading and writing of FITS images.

5.3.3.7 LVV-T2496 - Verify filename invariance

Version **1**. Open *LVV-T2496* test case in Jira.

Verify that for all datasets stored with unique filenames (or paths) as part of a Data Release, the name of the file retrieved by an external user is also unique and has a predictable name that is not dependent on data access mechanism

This behavior is not guaranteed by code, but it is the way we have configured our filename templates.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.8 LVV-T2495 - Verify Combining composite datasets for export

Version **1**. Open *LVV-T2495* test case in Jira.

Verify that a facility is available to combine file-based composite datasets into a single file in a Scientific Data Format

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Examine a ‘calexp‘ stored as a FITS file to confirm that its components (e.g., WCS, PSF, photocalib) are all contained within the same FITS file containing the image.

Expected Result

A FITs file with multiple extensions containing the various pieces of the composite ‘calexp‘ Dataset.

Actual Result

On lsst-devl, with Science Pipelines set up, open an ipython session, then do the following.

Choose an arbitrary ‘calexp‘ FITS file from the RC2 dataset in /repo/main:
In [**50**]: calexp_path = "/repo/main/HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z/calexp/20150116/z/HSC-Z/17926/"
In [**51**]: calexp_filename = "calexp_HSC_z_HSC-Z_17926_1_53_HSC_runs_RC2_w_2022_12_DM-34125_20220319T213338Z.fits"

Open this FITS file with the Astropy IO tools:
In [**52**]: **from astropy.io import** fits
In [**53**]: hdu = fits.open(calexp_path + calexp_filename)

Print the file's info to confirm that the FITS file contains multiple extensions with Datasets that make up the composite ‘calexp‘ dataset:
In [**54**]: hdu.info()
Filename: /repo/main/HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z/calexp/20150116/z/HSC-Z/17926/calexp_HSC_z_H

Z_17926_1_53_HSC_runs_RC2_w_2022_12_DM-34125_20220319T213338Z.fits
No. Name Ver Type Cards Dimensions Format
0 IMAGE 1 PrimaryHDU 216 ()
1 IMAGE 1 CompImageHDU 82 (2048, 4176) float32
2 MASK 1 CompImageHDU 92 (2048, 4176) int32
3 VARIANCE 1 CompImageHDU 82 (2048, 4176) float32
4 ARCHIVE_INDEX 1 BinTableHDU 41 313R x 7C [1J, 1J, 1J, 1J, 1J, 64A, 64A]
5 FilterLabel 1 BinTableHDU 28 1R x 3C [2X, 32A, 32A]
6 Detector 1 BinTableHDU 106 1R x 20C [1QA(4), 1J, 1J, 1QA(1), 1J, 1J, 1J, 1J, 1D, 1D, 1D, 1D, 1D, 1D, 1D, 1D, 1D, 1J, 1QE(0), 1QA(3)]
7 TransformMap 1 BinTableHDU 33 226R x 5C [1QA(10), 1QA(4), 1QA(18), 1QA(4), 1J]
8 ExposureSummaryStats 1 BinTableHDU 18 228R x 1C [1QB(13854)]
9 Detector 1 BinTableHDU 200 4R x 38C [3X, 1QA(1), 1J, 1J, 1J, 1J, 1D, 1D, 1D, 1D, 1J, 1QD(4), 1QA(7), 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1J, 1D, 1D, 1QA(1)]
10 ProductTransmissionCurve 1 BinTableHDU 21 5R x 2C [1J, 1J]
11 TransformedTransmissionCurve 1 BinTableHDU 21 1R x 2C [1J, 1J]
12 SpatiallyConstantTransmissionCurve 1 BinTableHDU 30 5R x 4C [1D, 1D, 1QD(1000), 1QD(1000)]
13 RadialTransmissionCurve 1 BinTableHDU 34 1R x 5C [1D, 1D, 1QD(10353), 1QD(357), 1QD(29)]
14 Polygon 1 BinTableHDU 21 8R x 2C [1D, 1D]
15 SkyWcs 1 BinTableHDU 17 1R x 1C [1QB(11112)]
16 PsfexPsf 1 BinTableHDU 52 1R x 9C [1J, 1J, 1J, 1J, 1J, 1J, 1D, 1D, 1E]
17 PsfexPsf 1 BinTableHDU 45 1R x 8C [2J, 1J, 6D, 6D, 3J, 39366E, 2D, 2D]
18 PhotoCalib 1 BinTableHDU 36 1R x 5C [1X, 1D, 1D, 1J, 1J]
19 ChebyshevBoundedField 1 BinTableHDU 41 32R x 6C [1J, 1J, 1J, 1J, 1J, 1D]
20 ApCorrMap 1 BinTableHDU 21 62R x 2C [64A, 1J]
21 ChebyshevBoundedField 1 BinTableHDU 41 31R x 6C [1J, 1J, 1J, 1J, 1J, 9D]

We have confirmed that the facility is available to serve composite datasets within a single scientific data file.

5.3.3.9 LVV-T2494 - Verify Strong exception guarantee

Version **1**. Open *LVV-T2494* test case in Jira.

Verify that a put operation on the Data Output System provides the strong exception guarantee. If a put operation fails the previous state shall be restored.

This is the usual behavior, and we regard it as a bug when it is violated, and we don't currently have any known bugs of this type.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute ButlerPutGetTests in https://github.com/lsst/daf_butler/blob/main/tests/test_butler.py

Expected Result

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_butler.py

Results:
tests/test_butler.py::PosixDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetConcrete PASSED

tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testBasicPutGet PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerMakeRepoOutfileTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetVirtual PASSED

All tests of the butler's "Put" and "Get" functionality passed. In particular, the 'runPutGetTests' section (lines 223-461 of that test suite, at the time of test execution) contains multiple instances of specifically testing whether 'butler.put()' fails when expected to, and then continuing on to do further 'butler.put()' actions after those failures, thus showing that state is maintained upon butler.put failure.

5.3.3.10 LVV-T2493 - Verify No clobber

Version **1**. Open *LVV-T2493* test case in Jira.

Verify that it is possible to configure the Data Output System such that it is an error to attempt to persist a dataset that is already present in the output repository

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute ButlerPutGetTests in https://github.com/lsst/daf_butler/blob/main/tests/test_butler.py

Expected Result

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_butler.py

Results:
tests/test_butler.py::PosixDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testBasicPutGet PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerMakeRepoOutfileTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testPutGet PASSED

tests/test_butler.py::S3DatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetVirtual PASSED

All tests of the butler's "Put" and "Get" functionality passed. In particular, the 'runPutGetTests' section that specifically demonstrates the failure when trying to 'butler.put' an existing dataset (lines 430-456 of that test suite, at the time of test execution) shows that an error is thrown when trying to persist a dataset that already exists in the repository.

5.3.3.11 LVV-T2492 - Verify Blocked write operation

Version **1**. Open *LVV-T2492* test case in Jira.

Verify that a put operation on the Data Output System blocks until it has either worked or failed

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description
Inspect relevant code in daf_butler

Expected Result

Actual Result

In lines 1081-1082 in daf/butler/_butler.py, the 'butler.put' method is defined as transactional, such that all database changes are reversed if there is an issue. Furthermore, datastore.put (also transactional) deletes the file if there is a problem after it has been written. Thus it is ensured that a put operation blocks until it has worked or failed.

5.3.3.12 LVV-T2491 - Verify Creation of new DatasetTypes

Version **1**. Open *LVV-T2491* test case in Jira.

Verify that the Data Output system allows a new DatasetType to be registered with a DataRepository, programmatically and at Supertask preflight-time, allowing Datasets of that DatasetType to be added to that DataRepository thereafter

Preconditions:

Execution status: **Pass**

Final comment:
Working on lsst-devl machines in a cloned 'daf_butler' repository at /project/jcarlin/SVV/gen3_middleware_ac

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute the unit tests in https://github.com/lsst/daf_butler/tests/test_sqlite.py and test_postgresql.py

Expected Result

Actual Result

Within daf_butler, execute:

'pytest -s -vv –no-header –cache-clear tests/test_sqlite.py'

The output from this command is very long, so we only show the final "summary":

335 passed, **8 skipped**, **3 warnings** in 176.68s (0:02:56)

Many of the unit tests contained in this suite are subclasses of those in 'python/lsst/daf/butler/registry/tests/_registry.py', which includes tests of registering new DatasetTypes.

We also executed 'pytest -s -vv –no-header –cache-clear tests/test_postgresql.py', in which all unit tests also passed. We have thus demonstrated that DatasetTypes can be registered with a DataRepository.

5.3.3.13 LVV-T2488 - Verify access outputs from test processing runs

Version **1**. Open *LVV-T2488* test case in Jira.

Verify that the Data Input System shall provide access to processing runs initiated for test/development purposes, from the same compute environment in which the processing was run

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Instantiate a Butler at NCSA or SLAC targeting a test run in '/repo/main'.

Expected Result

Actual Result

On lsst-devl machines at NCSA, with Science Pipelines set up:

Open an ipython session:

$ ipython

Import the butler, initialize it for a collection containing RC2 processing:

In [**1**]: **from lsst.daf.butler import** Butler
In [**2**]: butler = Butler("/repo/main", collections=["HSC/runs/RC2/w_2022_12/DM-34125"])

Select a random dataId:
In [**3**]: dataid = {"visit": 1230, "instrument": "HSC", "detector": 43}

Step 2	Step Execution Status: **Pass**

Description

Call Butler.get.

Expected Result

Actual Result

Retrieve the 'calexp' and its associated 'wcs':
In [**4**]: calexp = butler.get("calexp", dataId=dataid)
In [**5**]: wcs = butler.get("calexp.wcs", dataId=dataid)

Step 3 Step Execution Status: **Pass**

Description

Verify that data is correctly retrieved

Expected Result

Actual Result

Examine the 'calexp' and 'wcs' to confirm that they are different:

In [**6**]: calexp

Out[**6**]: <lsst.afw.image.exposure.ExposureF at 0x7fc6d9b83130>

As expected, the calexp is an ExposureF object.

In [**7**]: wcs

Out[**7**]:

FITS standard SkyWcs:

Sky Origin: (149.8520271457, +2.0585702399)

Pixel Origin: (1003.05, 2415.24)

Pixel Scale: 0.16713 arcsec/pixel

The WCS looks like a properly defined WCS. Now look at the image plane of the calexp:

In [**8**]: calexp.image

Out[**8**]:

lsst.afw.image.image.ImageF=[[-0.36441362 -0.3609193 -0.35746038 ... -25.336197 -25.346905
-25.357626]
[-0.3578999 -0.354396 -0.3511718 ... -25.327019 -25.337673
-25.348345]
[-0.3513667 -0.34809738 -0.34461957 ... -25.317785 -25.328388
-25.339252]
...
[28.878033 28.84473 28.811472 ... 7.566758 7.5311904
7.4954834]
[28.914822 28.88162 28.848219 ... 7.5671864 7.5316124
7.4958982]
[28.951662 28.918072 28.884766 ... 7.567666 7.5320854

7.4963655]], bbox=(minimum=(0, 0), maximum=(2047, 4175))

These look good. We have thus demonstrated that the data products of test processing performed on the development machines at NCSA can be retrieved on those same machines.

5.3.3.14 LVV-T2487 - Verify Accessing official Data Releases

Version **1**. Open *LVV-T2487* test case in Jira.

Verify that the Data Input System interface shall provide access to official Data Releases from the LSST Science Platform.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Instantiate a butler on RSP targeting DP0.x collections.

Expected Result

Actual Result

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

Description
Call 'Butler.get'

Expected Result

Actual Result

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

Description
Verify that data is retrieved

Expected Result

Actual Result

5.3.3.15 LVV-T2486 - Verify Consistent input interface

Version **1**. Open *LVV-T2486* test case in Jira.

Verify that the Data Input System provides a consistent interface for loading Datasets into memory given a DatasetRef across different types of DataRepositories

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Run a 'PipelineTask' against a local SQLite+POSIX repo

Expected Result

Actual Result

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

Description

Run the same 'PipelineTask' against a PostgreSQL+POSIX repo.

Expected Result

Actual Result

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

Description

Run the same 'PipelineTask' against a PostgreSQL+S3 repo.

Expected Result

Actual Result

5.3.3.16 LVV-T2485 - Verify Local caching of remote resources

Version **1**. Open *LVV-T2485* test case in Jira.

Verify that it is possible to configure the Data Input System to cache a local version of a Dataset that has been retrieved from a remote DataRepository.

Note that this doesn't really look distinct from DMS-MWBT-REQ-0055 anymore; I think 0055 was perhaps supposed to be some kind of shared-filesystem proxy for something that lives on even slower storage, like tape.
The specs are similar enough that the same test can be used

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description
Enable datastore caching in a Butler client in RSP (or any S3-backed repo).

Expected Result

Actual Result

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

Description

Run butler.get twice, check (e.g. trace logs) that the second comes from

Expected Result

Actual Result

5.3.3.17 LVV-T2484 - Verify Local proxy

Version **1**. Open *LVV-T2484* test case in Jira.

Verify that it is possible to configure the Data Input system to use a local proxy to share remote retrievals of common Datasets

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Enable datastore caching in a Butler client in RSP (or any S3-backed repo).

Expected Result

Actual Result

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

Description

Run 'butler.get' twice, check (e.g. trace logs) that the second comes from

Expected Result

Actual Result

5.3.3.18 LVV-T2483 - Verify Failure on missing input file

Version **1**. Open *LVV-T2483* test case in Jira.

Verify that it is possible via configuration to require the Data Input System to fail if an expected file is not found at the specified location

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Manually create QG with execution butler.

Expected Result

Actual Result

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

Description

Run 'butler.get' against execution butler for ref that does not exist.

Expected Result

Actual Result

5.3.3.19 LVV-T2482 - Verify Enabling PipelineTasks to execute

Version **1**. Open *LVV-T2482* test case in Jira.

Verify that it is possible for the Data Input System to construct an InMemoryDataset from a set of files stored locally on disk (without a remote database connection)

Preconditions:

Execution status: **In Progress**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Manually create QG with execution butler

Expected Result

Actual Result

Create the quantum graph and execution butler for three visits from tract 9813 in HSC RC2 data located in /repo/-main at NCSA.

pipetask qgraph -b /repo/main/butler.yaml -p $OBS_SUBARU_DIR/pipelines/DRP.yaml#step1
-i HSC/runs/RC2/w_2022_12/DM-34125 -o u/jcarlin/qgraph_test_LDM556 -d “visit in (1230, 1232, 1240)”
-q test.qgraph –save-execution-butler test_execution_butler

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

Description

Run ‘butler.get‘ against execution butler.

Expected Result

Actual Result

cd test_execution_butler/

5.3.3.20 LVV-T2481 - Verify third party datasets

Version **1**. Open *LVV-T2481* test case in Jira.

Verify that it is possible for the Data Input System to read from catalogs provided by outside sources using the same interface used for reading first class LSST datasets via a different plugin.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Make an empty repo.

Expected Result

Actual Result

With the science pipelines set up, open an ipython session, then:

In [1]: from lsst.daf.butler import Butler

Rather than making an empty repo, we'll create a new run in an existing butler repository.

In [2]: butler = Butler("/project/jcarlin/repos/rc2_subset/SMALL_HSC", run="testrun"))

Step 2 Step Execution Status: **Pass**

Description

Ingest some external parquet or FITS catalog.

Expected Result

Actual Result

In [4]: from astropy.io import ascii

Ingest a CSV catalog that was obtained by searching the WISE catalog via the IRSA archive, selecting objects within 30 arcsec of MESSIER 033, and saving the results to a CSV file.

In [5]: tab = ascii.read("table_irsa_catalog_search_results.csv")

In [6]: tab
Out[6]:
<Table length=8>
designation ra dec sigra sigdec sigradec w1mpro w1sigmpro w1snr w1rchi2 ... w1nm w1m w2nm w2m w3nm w3m w4nm w4m dist angle
str19 float64 float64 float64 float64 float64 float64 float64 float64 float64 ... int64 int64 int64 int64 int64 int64 int64 int64 float64 float64
——————- ———- ———- ——- ——- ——– ——- ——— ——- ——- ... —– —– —– —– —– —– —– —– ——— ———-
J013350.89+303936.7 23.462066 30.660195 0.045 0.044 0.0024 10.639 0.025 44.3 14.95 ... 35 35 35 35 24 24 22 24 0.120773 138.183704
J013351.40+303953.2 23.4641735 30.6647787 0.0821 0.086 -0.0237 12.685 0.048 22.4 17.67 ... 35 35 35 35 24 24 24 24 17.691256 21.927817
J013352.24+303942.9 23.4676801 30.6619338 0.0832 0.083 -0.0145 12.574 0.043 25.3 22.31 ... 35 35 35 35 5 24 19 24 18.523464 70.543114
J013351.32+303956.0 23.4638748 30.6655762 0.0842 0.0886 -0.022 12.7 0.051 21.2 16.24 ... 35 35 35 35 24 24 24 24 20.101993 16.417721
J013349.51+303956.4 23.4563181 30.6656714 0.1087 0.1185 -0.0375 13.18 0.125 8.7 2.402 ... 35 35 35 35 24 24 24 24 26.440434 317.923691
J013352.48+303954.9 23.4686874 30.6652703 0.1379 0.1404 -0.0365 13.036 0.059 18.3 11.07 ... 36 36 35 36 2 24

0 24 27.464228 48.546387
J013350.07+303911.3 23.4586394 30.6531485 0.1934 0.2011 -0.054 12.686 0.081 13.5 3.44 ... 35 35 35 35 0 22 0 24 27.549678 202.47468
J013349.03+303951.5 23.4543307 30.6643274 0.1148 0.1222 -0.0355 13.115 0.12 9.1 2.649 ... 35 35 35 35 24 24 24 24 28.081582 301.775292

Step 3 Step Execution Status: **Pass**

Description

Call butler put

Expected Result

Actual Result

Create a dummy dataId:
In [7]: dataId = {"instrument": "WISE", "visit": 423}

Register the datasetType:
In [8]: from lsst.daf.butler import DatasetType
In [9]: datasetType = DatasetType("table", [], "AstropyTable", universe=butler.registry.dimensions)
In [10]: butler.registry.registerDatasetType(datasetType)
Out[10]: True

Now use 'butler.put' to put the table into the repo:
In [11]: ref = butler.put(tab, datasetType)

Retrieve the table we just put into the repo:
In [12]: uri = butler.getURI(ref)
In [13]: table = butler.get("table")

Confirm that we get back the same table we started with:
In [14]: table
Out[14]:

<Table length=8>
designation ra dec sigra sigdec sigradec w1mpro w1sigmpro w1snr w1rchi2 ... w1nm w1m w2nm w2m w3nm w3m w4nm w4m dist angle
str19 float64 float64 float64 float64 float64 float64 float64 float64 float64 ... int64 int64 int64 int64 int64 int64 int64 int64 float64 float64
——————- ———- ———- ——- ——- ——– ——- ——— ——- ——- ... —- —- —- —- —- —- —- —- ————— ———-
J013350.89+303936.7 23.462066 30.660195 0.045 0.044 0.0024 10.639 0.025 44.3 14.95 ... 35 35 35 35 24 24 22 24 0.120773 138.183704
J013351.40+303953.2 23.4641735 30.6647787 0.0821 0.086 -0.0237 12.685 0.048 22.4 17.67 ... 35 35 35 35 24 24 24 24 17.691256 21.927817
J013352.24+303942.9 23.4676801 30.6619338 0.0832 0.083 -0.0145 12.574 0.043 25.3 22.31 ... 35 35 35 35 5 24 19 24 18.523464 70.543114
J013351.32+303956.0 23.4638748 30.6655762 0.0842 0.0886 -0.022 12.7 0.051 21.2 16.24 ... 35 35 35 35 24 24 24 24 20.101993 16.417721
J013349.51+303956.4 23.4563181 30.6656714 0.1087 0.1185 -0.0375 13.18 0.125 8.7 2.402 ... 35 35 35 35 24 24 24 24 26.440434 317.923691
J013352.48+303954.9 23.4686874 30.6652703 0.1379 0.1404 -0.0365 13.036 0.059 18.3 11.07 ... 36 36 35 36 2 24 0 24 27.464228 48.546387
J013350.07+303911.3 23.4586394 30.6531485 0.1934 0.2011 -0.054 12.686 0.081 13.5 3.44 ... 35 35 35 35 0 22 0 24 27.549678 202.47468
J013349.03+303951.5 23.4543307 30.6643274 0.1148 0.1222 -0.0355 13.115 0.12 9.1 2.649 ... 35 35 35 35 24 24 24 24 28.081582 301.775292

5.3.3.21 LVV-T2480 - Verify Item from Composite Datasets

Version **1**. Open *LVV-T2480* test case in Jira.

Verify that it is possible to load into memory an item from a Composite Dataset without loading the full Dataset.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Use a butler to read a component (e.g. WCS), against any repo.

Expected Result

Actual Result

On lsst-devl machines at NCSA, with Science Pipelines set up:

Open an ipython session:

$ ipython

Import the butler, initialize it for a collection containing RC2 processing:

In [**1**]: **from lsst.daf.butler import** Butler
In [**2**]: butler = Butler("/repo/main", collections=["HSC/runs/RC2/w_2022_12/DM-34125"])

Select a random dataId:
In [**3**]: dataid = {"visit": 1230, "instrument": "HSC", "detector": 43}

Retrieve the 'calexp' and its associated 'wcs':
In [**4**]: calexp = butler.get("calexp", dataId=dataid)
In [**5**]: wcs = butler.get("calexp.wcs", dataId=dataid)

Examine the 'calexp' and 'wcs' to confirm that they are different:
In [**6**]: calexp
Out[**6**]: <lsst.afw.image.exposure.ExposureF at 0x7f8db02f4f30>

In [**7**]: wcs

Out[**7**]:
FITS standard SkyWcs:
Sky Origin: (149.8520271457, +2.0585702399)
Pixel Origin: (1003.05, 2415.24)
Pixel Scale: 0.16713 arcsec/pixel

They clearly differ (the 'calexp' is an ExposureF type object, while the 'wcs' is a FITS standard SkyWcs). Now confirm that the 'wcs' is an item contained within the composite 'calexp' dataset:

In [**8**]: calexp.wcs
Out[**8**]:
FITS standard SkyWcs:
Sky Origin: (149.8520271457, +2.0585702399)
Pixel Origin: (1003.05, 2415.24)
Pixel Scale: 0.16713 arcsec/pixel

This is identical to the 'wcs' retrieved above, thus demonstrating that the 'wcs' item contained within the composite 'calexp' dataset can be retrieved alone, without loading the entire 'calexp'.

5.3.3.22 LVV-T2479 - Verify Parameterized subset of a Dataset

Version **1**. Open *LVV-T2479* test case in Jira.

Verify that It is possible to load into memory a parameterized subset of a Dataset without loading the full Dataset.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Use a butler to read a subimage via get parameters against any repo.

Expected Result

Actual Result

On lsst-devl machines at NCSA, with Science Pipelines set up:

Open an ipython session:

$ ipython

Import the butler, initialize it for a collection containing RC2 processing:

In [**1**]: **from lsst.daf.butler import** Butler
In [**2**]: butler = Butler("/repo/main", collections=["HSC/runs/RC2/w_2022_12/DM-34125"])

Select a random dataId:
In [**3**]: dataid = {"visit": 1230, "instrument": "HSC", "detector": 43}

Retrieve the 'calexp' and its associated 'wcs':
In [**4**]: calexp = butler.get("calexp", dataId=dataid)
In [**5**]: wcs = butler.get("calexp.wcs", dataId=dataid)

What are the dimensions of this calexp?
In [**32**]: calexp.getDimensions()
Out[**32**]: Extent2I(2048, 4176)

Look up the sky origin of this'calexp':
In [**33**]: wcs.getSkyOrigin()
Out[**33**]: SpherePoint(149.85202714570465*degrees, 2.058570239877529*degrees)

Now, import the "geom" package, and create a position slightly shifted from the sky origin on which to center a cutout image:
In [**41**]: **import lsst.geom as geom**
In [**42**]: pos = geom.SpherePoint(149.8525, 2.06, geom.degrees)

Extract a cutout image with extent of 140 pixels:
In [**43**]: cutout = butler.get("calexp", dataId=dataid).getCutout(pos, geom.Extent2I(140))

Confirm that we have obtained an image that is smaller than the original 'calexp':
In [**44**]: cutout.getDimensions()
Out[**44**]: Extent2I(140, 140)

We have thus demonstrated that a parameterized subset (in this case, a sub-image) of a Dataset can be loaded into memory without loading the full Dataset.

5.3.3.23 LVV-T2478 - Verify I/O using cloud storage

Version **1**. Open *LVV-T2478* test case in Jira.

Verify that the Data Input/Output System shall be able to utilize cloud-based storage engines.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Make an empty repo with an S3 datastore,

Expected Result

Actual Result

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

Description

Run 'butler get/put'

Expected Result

Actual Result

5.3.3.24 LVV-T2477 - Verify I/O using distributed file system

Version **1**. Open *LVV-T2477* test case in Jira.

Verify that the Data Input/Output System shall be able to read/write from/to distributed file systems.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Make an empty repo with a POSIX datastore

Expected Result

Actual Result

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

Description

do butler get/put

Expected Result

Actual Result

5.3.3.25 LVV-T2476 - Verify Format Plugability

Version **1**. Open *LVV-T2476* test case in Jira.

Verify that it is possible to control the method used to read and write a particular DatasetType using a text configuration file such that the Python object and the form of the persisted dataset can be configured externally.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Make an empty repo with the default configuration.

Expected Result

Actual Result

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

Description

Make an empty repo with configuration that overrides a formatter.

Expected Result

Actual Result

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

Description

Make an empty repo with configuration that changes a StorageClass's Python

Expected Result

Actual Result

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

Description

Put and get the same datasets to all repos.

Expected Result

Actual Result

5.3.3.26 LVV-T2474 - Verify Data Discovery for Data Release Production

Version **1**. Open *LVV-T2474* test case in Jira.

Verify that the Data Discovery System interface is usable when initiating processing for Data Release Production.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Run QG generation for the DRP pipeline against any major repo (e.g. '/repo/main'). Same as for LVV-T2473

Expected Result

Actual Result

At NCSA, on lsst-devl machines, with science pipelines set up.

Generate a quantum graph by executing "step1" of the standard DRP pipeline against a recent reprocessing of HSC RC2 data:

pipetask qgraph -b /repo/main/butler.yaml -p $OBS_SUBARU_DIR/pipelines/DRP.yaml#step1 -i HSC/runs/RC2/w_2022_12/DM-34125 -o u/jcarlin/qgraph_test_LDM556_step1 -d "visit in (1230, 1232) AND detector in (42, 43)" -q rc2_step1.qgraph

This prepares the quantum graph illustrating "step1" processing of two visits and two detectors from the RC2 dataset, and saves the graph as "rc2_step1.qgraph".

To illustrate that the graph is well-formed, examine the "graph" output by executing:
pipetask qgraph -b /repo/main/butler.yaml -g rc2_step1.qgraph –show graph

Outputs:
TaskDef(lsst.ip.isr.isrTask.IsrTask, label=isr)
Quantum 0:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []
DatasetType('isrOverscanCorrected', {band, instrument, detector, physical_filter, exposure}, Exposure): []
DatasetType('bias', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('transmission_filter', {band, instrument, physical_filter}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', physical_filter: 'HSC-I', ...})]
DatasetType('bfKernel', {instrument}, NumpyArray, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_optics', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]

DatasetType('crosstalk', {instrument, detector}, CrosstalkCalib, isCalibration=True): []
DatasetType('fringe', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): []
DatasetType('transmission_sensor', {instrument, detector}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('flat', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43, physical_filter: 'HSC-I', ...})]
DatasetType('camera', {instrument}, Camera, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_atmosphere', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('dark', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('yBackground', {band, instrument, detector, physical_filter}, StrayLightData, isCalibration=True): []
DatasetType('defects', {instrument, detector}, Defects, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('brighterFatterKernel', {instrument, detector}, BrighterFatterKernel, isCalibration=True): []
outputs:
DatasetType('postISRCCD', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_metadata', {band, instrument, detector, physical_filter, exposure}, PropertySet): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_log', {band, instrument, detector, physical_filter, exposure}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
Quantum 1:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1232, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []

...

TaskDef(lsst.pipe.tasks.postprocess.WriteSourceTableTask, label=writeSourceTable)
Quantum 0:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]

Quantum 1:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
Quantum 2:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
Quantum 3:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]

(truncated for display) This shows the quanta and the tasks to be executed for each. This graph file demonstrates that the Data Discovery System can be used to initiate Data Release Production runs.

5.3.3.27 LVV-T2475 - Verify Data discovery for test processing runs

Version **1**. Open *LVV-T2475* test case in Jira.

Verify that the Data Discovery System interface is usable when initiating processing runs initiated for test/development purposes (on LSST or personal hardware),

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Run QG generation for the DRP pipeline against any major repo (e.g. '/repo/main'). Same as for LVV-T2473

Expected Result

Actual Result

At NCSA, on lsst-devl machines, with science pipelines set up.

Generate a quantum graph by executing "step1" of the standard DRP pipeline against a recent reprocessing of HSC RC2 data:

pipetask qgraph -b /repo/main/butler.yaml -p $OBS_SUBARU_DIR/pipelines/DRP.yaml#step1 -i HSC/runs/RC2/w_2022_12/DM-34125 -o u/jcarlin/qgraph_test_LDM556_step1 -d "visit in (1230, 1232) AND detector in (42, 43)" -q rc2_step1.qgraph

This prepares the quantum graph illustrating "step1" processing of two visits and two detectors from the RC2 dataset, and saves the graph as "rc2_step1.qgraph".

To illustrate that the graph is well-formed, examine the "graph" output by executing:
pipetask qgraph -b /repo/main/butler.yaml -g rc2_step1.qgraph –show graph

Outputs:
TaskDef(lsst.ip.isr.isrTask.IsrTask, label=isr)
Quantum 0:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []
DatasetType('isrOverscanCorrected', {band, instrument, detector, physical_filter, exposure}, Exposure): []
DatasetType('bias', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('transmission_filter', {band, instrument, physical_filter}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', physical_filter: 'HSC-I', ...})]
DatasetType('bfKernel', {instrument}, NumpyArray, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_optics', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('crosstalk', {instrument, detector}, CrosstalkCalib, isCalibration=True): []
DatasetType('fringe', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): []
DatasetType('transmission_sensor', {instrument, detector}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('flat', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43, physical_filter: 'HSC-I', ...})]
DatasetType('camera', {instrument}, Camera, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_atmosphere', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('dark', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('yBackground', {band, instrument, detector, physical_filter}, StrayLightData, isCalibration=True): []
DatasetType('defects', {instrument, detector}, Defects, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('brighterFatterKernel', {instrument, detector}, BrighterFatterKernel, isCalibration=True): []
outputs:
DatasetType('postISRCCD', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_metadata', {band, instrument, detector, physical_filter, exposure}, PropertySet): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_log', {band, instrument, detector, physical_filter, exposure}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
Quantum 1:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1232, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []

...

TaskDef(lsst.pipe.tasks.postprocess.WriteSourceTableTask, label=writeSourceTable)
Quantum 0:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
Quantum 1:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
Quantum 2:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
Quantum 3:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords):

[DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]

(truncated for display) This shows the quanta and the tasks to be executed for each. This graph file demonstrates that the Data Discovery System can be used to initiate test/development processing runs.

5.3.3.28 LVV-T2473 - Verify Consistent discovery interface

Version **1**. Open *LVV-T2473* test case in Jira.

Verify that the Data Discovery System provides a consistent interface for obtaining a graph that represents the DataUnits and Datasets in a DataRepository that match user specified criteria.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Run QG generation against any major repo (e.g. '/repo/main').

Expected Result

Actual Result

At NCSA, on lsst-devl machines, with science pipelines set up.

Generate a quantum graph by executing "step1" of the standard DRP pipeline against a recent reprocessing of HSC RC2 data:

pipetask qgraph -b /repo/main/butler.yaml -p $OBS_SUBARU_DIR/pipelines/DRP.yaml#step1 -i HSC/runs/RC2/w_2022_12/DM-34125 -o u/jcarlin/qgraph_test_LDM556_step1 -d "visit in (1230, 1232) AND detector in (42, 43)" -q rc2_step1.qgraph

This prepares the quantum graph illustrating "step1" processing of two visits and two detectors from the RC2 dataset, and saves the graph as "rc2_step1.qgraph".

To illustrate that the graph is well-formed, examine the "graph" output by executing:
pipetask qgraph -b /repo/main/butler.yaml -g rc2_step1.qgraph –show graph

Outputs:
TaskDef(lsst.ip.isr.isrTask.IsrTask, label=isr)
Quantum 0:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []
DatasetType('isrOverscanCorrected', {band, instrument, detector, physical_filter, exposure}, Exposure): []
DatasetType('bias', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('transmission_filter', {band, instrument, physical_filter}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', physical_filter: 'HSC-I', ...})]
DatasetType('bfKernel', {instrument}, NumpyArray, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_optics', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('crosstalk', {instrument, detector}, CrosstalkCalib, isCalibration=True): []
DatasetType('fringe', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): []
DatasetType('transmission_sensor', {instrument, detector}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('flat', {band, instrument, detector, physical_filter}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43, physical_filter: 'HSC-I', ...})]
DatasetType('camera', {instrument}, Camera, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('transmission_atmosphere', {instrument}, TransmissionCurve, isCalibration=True): [DataId({instrument: 'HSC'})]
DatasetType('dark', {instrument, detector}, ExposureF, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('yBackground', {band, instrument, detector, physical_filter}, StrayLightData, isCalibration=True): []

DatasetType('defects', {instrument, detector}, Defects, isCalibration=True): [DataId({instrument: 'HSC', detector: 43})]
DatasetType('brighterFatterKernel', {instrument, detector}, BrighterFatterKernel, isCalibration=True): []
outputs:
DatasetType('postISRCCD', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_metadata', {band, instrument, detector, physical_filter, exposure}, PropertySet): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
DatasetType('isr_log', {band, instrument, detector, physical_filter, exposure}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, exposure: 1230, ...})]
Quantum 1:
inputs:
DatasetType('raw', {band, instrument, detector, physical_filter, exposure}, Exposure): [DataId({instrument: 'HSC', detector: 43, exposure: 1232, ...})]
DatasetType('linearizer', {instrument, detector}, Linearizer, isCalibration=True): []

...

TaskDef(lsst.pipe.tasks.postprocess.WriteSourceTableTask, label=writeSourceTable)
Quantum 0:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, visit: 1232, ...})]
Quantum 1:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1230, ...})]
Quantum 2:

inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 42, visit: 1232, ...})]
Quantum 3:
inputs:
DatasetType('src', {band, instrument, detector, physical_filter, visit_system, visit}, SourceCatalog): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
outputs:
DatasetType('writeSourceTable_metadata', {band, instrument, detector, physical_filter, visit_system, visit}, PropertySet): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('source', {band, instrument, detector, physical_filter, visit_system, visit}, DataFrame): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]
DatasetType('writeSourceTable_log', {band, instrument, detector, physical_filter, visit_system, visit}, ButlerLogRecords): [DataId({instrument: 'HSC', detector: 43, visit: 1230, ...})]

(truncated for display) This shows the quanta and the tasks to be executed for each. This graph file demonstrates that the requested Datasets and DataUnits have been compiled into the serialized quantum graph.

5.3.3.29 LVV-T2472 - Verify Introspection for DatasetExpressions

Version **1**. Open *LVV-T2472* test case in Jira.

Verify that the Data Discovery System allows for a DatasetExpression to be constructed interactively using introspection on the DataRepository schema

Note that the requirement talks about high-level interactive tooling, but description makes it clear that middleware is only responsible for exposing the introspection necessary to allow that tooling to be written, and we do.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Print dimension metadata schema by walking through DimensionUniverse.

Expected Result

Actual Result

5.3.3.30 LVV-T2471 - Verify Filter by non-DatasetRef Database Entries

Version **1**. Open *LVV-T2471* test case in Jira.

Verify that the Data Discovery System is able to filter search results based upon specified filters that need non-DatasetRef database entries

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Run 'butler query-datasets' against any major repo (e.g. '/repo/main'), with a WHERE expression involving some dimension metadata fields.

Expected Result

Actual Result

We demonstrate this using the same query as in LVV-T2467. This query includes a "WHERE" clause to select 'calexp' datasets based on tract and patch criteria. Because tract/patch are not dimensions of a 'calexp', this demonstrates the use of dimension metadata for selection.

butler query-datasets /repo/main calexp –where "tract=9615 AND patch=43 AND skymap='hsc_rings_v1'" –collections HSC/runs/RC2/w_2022_12/DM-34125 | less

The first few lines of the returned table are captured in this screenshot:

type	run	id	band	instrument	detector	physical_filter	visit_system	visit
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	2fe96249-9cdc-4ebc-a596-14f527d700bf	y	HSC	24	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	cb3a756c-c511-4e36-a846-650880223aa7	y	HSC	25	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	d1fb78c2-3419-49cc-86a3-3dc2a0db7958	y	HSC	26	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	5733682f-3ccf-461a-9a83-be68e84a4d68	y	HSC	32	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	7d7d3084-2dbd-4ea8-8f40-068341898f48	y	HSC	33	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	4dc43ff7-df44-4f44-a44b-5ab4d75ea0fb	y	HSC	34	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	29cd5478-f3a2-4993-a6ef-8bb0e9455ce5	y	HSC	40	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	904e00c8-578a-45bd-a453-ccb0772dce49	y	HSC	41	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	f5a7c106-840f-4f04-ab84-e55126832162	y	HSC	42	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	8a95c42c-87fe-4b43-8c60-a9031b7d414e	y	HSC	91	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	71bce3cb-2c6e-41e9-bdaa-5f75a91b8bc0	y	HSC	92	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	1d7ced69-8869-4dfb-a4b7-e00e496e6814	y	HSC	96	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	caec1cf2-7ed0-4629-a6bf-feaf4ab7fed0	y	HSC	97	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	ddeb90af-f966-4951-b01e-a886512e5c1f	y	HSC	62	HSC-Y	0	440

This demonstrates the use of dimension metadata for filtering search results from the butler.

5.3.3.31 LVV-T2470 - Verify Dataset overrides

Version **1**. Open *LVV-T2470* test case in Jira.

Verify that it is possible for an operator to configure the Data Discovery System to override certain Datasets with others before retrieval.

Preconditions:

Execution status: **Pass**

Final comment:
We verify this with the same query as used in LVV-T2469, but instead specifying "findFirst=True" to override the default behavior.

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Run 'butler query-datasets' against any major repo (e.g. '/repo/main'), with multiple input collections that contain the same unresolved DatasetRefs, with findFirst=True.

Expected Result

Actual Result
butler query-datasets /repo/main deepCoadd_calexp –collections HSC/runs/RC2/w_2022_12/DM-34125,HSC/runs/RC2/w_2022_08/I
33741 –where "tract=9615 AND patch=43 AND band='i' AND skymap='hsc_rings_v1'" –find-first

type	run	id	band	skymap	tract	patch
deepCoadd_calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z	f7ff9ce8-ef64-4d8e-a4f3-fda17335388a	i	hsc_rings_v1	9615	43

As expected, this query returns a single result – the "first."

5.3.3.32 LVV-T2469 - Verify Multiple parallel input Collections

Version **1**. Open *LVV-T2469* test case in Jira.

Verify that the Data Discovery System is able to locate Datasets from multiple input Collections in order to retrieve the same logical Dataset from them all.

This is to allow for comparison of the same data reduced with multiple different stacks.

Preconditions:

Execution status: **Pass**

Final comment:
We verify this by demonstrating that a 'deepCoadd_calexp' can be retrieved for the same tract, patch, band combination, but from different collections (i.e., data processed with different pipeline versions).

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Run 'butler query-datasets' against any major repo (e.g. '/repo/main'), with multiple input collections that contain the same unresolved DatasetRefs, with findFirst=False

Expected Result

Actual Result
butler query-datasets /repo/main deepCoadd_calexp –collections HSC/runs/RC2/w_2022_12/DM-34125,HSC/runs/RC2/w_2022_08/DM-33741 –where "tract=9615 AND patch=43 AND band='i' AND skymap='hsc_rings_v1'"

type	run	id	band	skymap	tract	patch
deepCoadd_calexp	HSC/runs/RC2/w_2022_08/DM-33741/20220222T202737Z	6df59532-dfbc-43c0-96c6-ecd58af6433c	i	hsc_rings_v1	9615	43
deepCoadd_calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z	f7ff9ce8-ef64-4d8e-a4f3-fda17335388a	i	hsc_rings_v1	9615	43

This demonstrates that the butler can identify the same logical Dataset ('deepCoadd_calexp') from different collections in the same query.

5.3.3.33 LVV-T2468 - Verify Multiple chained input Collections

Version **1**. Open *LVV-T2468* test case in Jira.

Verify that the Data Discovery System is able treat multiple input Collections as a single coherent logical repository

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Run 'butler query-datasets' against any major repo (e.g 'repo/main') with multiple input collections.

Expected Result

Actual Result

This will be demonstrated by showing that datasets of type 'objectTable_tract' can be retrieved by a butler query from multiple collections.

Execute the following query to retrieve 'objectTable_tract' from RC2 reprocessing collections corresponding to weekly pipelines from w_2022_08 and w_2022_12:

butler query-datasets /repo/main objectTable_tract –collections HSC/runs/RC2/w_2022_12/DM-34125,HSC/runs/RC2/w_2022_08/DM-33741

This returns the following table:

type	run	id	skymap	tract
objectTable_tract	HSC/runs/RC2/w_2022_08/DM-33741/20220222T202737Z	e19f9def-3d4a-4fd1-b7f1-ebbc53cb74ea	hsc_rings_v1	9615
objectTable_tract	HSC/runs/RC2/w_2022_08/DM-33741/20220222T202737Z	dec24ba7-0e8f-467c-b609-281abadf2378	hsc_rings_v1	9697
objectTable_tract	HSC/runs/RC2/w_2022_08/DM-33741/20220222T202737Z	fba3b879-57f4-4f35-8c56-daee1c65f5cf	hsc_rings_v1	9813
objectTable_tract	HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z	f97c9384-826c-4306-86c6-a6250d68271f	hsc_rings_v1	9615
objectTable_tract	HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z	d3f6cc84-782e-4d1b-975b-715798124e33	hsc_rings_v1	9697
objectTable_tract	HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z	f4877483-d603-4c53-9a42-cd5d5fa7b5d8	hsc_rings_v1	9813

We have thus demonstrated that datasets from multiple collections can be retrieved by the butler as a single coherent unit.

5.3.3.34 LVV-T2466 - Verify enable complete pipeline specification

Version **1**. Open *LVV-T2466* test case in Jira.

Verify that the design provides an interface for delivering a complete algorithmic work specification (a “Pipeline specification”) from Science Pipelines to an execution system, the “supervisory framework”, a notable instance of which is the LSST production system.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

This is a fundamental part of the design of PipelineTask.

Expected Result

Actual Result

5.3.3.35 LVV-T2467 - Verify DataUnit lookup: processing driven

Version **1**. Open *LVV-T2467* test case in Jira.

Verify that all Data Discovery Systems make it possible to discover the DataUnits for all Datasets that could potentially be used to produce a given DatasetType with known DataUnits.

Preconditions:

Execution status: **Pass**

Final comment:
We will verify this by demonstrating that all dataset overlapping a given tract/patch combination (and thus a specific sky region) can be readily discovered.

Detailed steps results:

Step 1	Step Execution Status: **Pass**

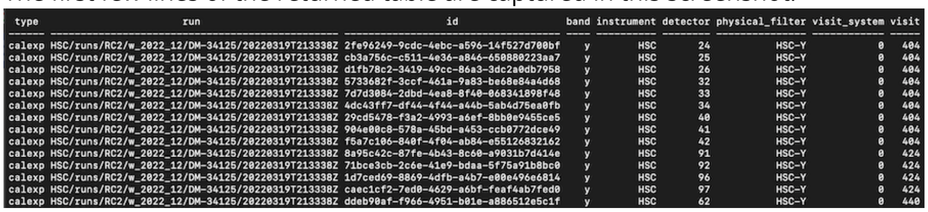
Description
Run 'butler query-datasets' against any major repo (e.g. '/repo/main').

Expected Result

Actual Result
This query returns a list of all 'calexp' datasets overlapping an arbitrarily chosen tract (known a priori to contain data in the HSC RC2 dataset), patch combination (9615, 43).

butler query-datasets /repo/main calexp –where "tract=9615 AND patch=43 AND skymap='hsc_rings_v1'" –collections HSC/runs/RC2/w_2022_12/DM-34125 | less

The first few lines of the returned table are captured in this screenshot:



type	run	id	band	instrument	detector	physical_filter	visit_system	visit
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	2fe96249-9cdc-4ebc-a596-14f527d700bf	y	HSC	24	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	cb3a756c-c511-4e36-a846-650880223aa7	y	HSC	25	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	d1fb78c2-3419-49cc-86a3-3dc2a0db7958	y	HSC	26	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	5733682f-3ccf-461a-9a83-be68e84a4d68	y	HSC	32	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	7d7d3084-2dbd-4ea8-8f40-068341898f48	y	HSC	33	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	4dc43ff7-df44-4f44-a44b-5ab4d75ea0fb	y	HSC	34	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	29cd5478-f3a2-4993-a6ef-8bb0e9455ce5	y	HSC	40	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	904e00c8-578a-45bd-a453-ccb0772dce49	y	HSC	41	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	f5a7c106-840f-4f04-ab84-e55126832162	y	HSC	42	HSC-Y	0	404
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	8a95c42c-87fe-4b43-8c60-a9031b7d414e	y	HSC	91	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	71bce3cb-2c6e-41e9-bdaa-5f75a91b8bc0	y	HSC	92	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	1d7ced69-8869-4dfb-a4b7-e00e496e6814	y	HSC	96	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	caec1cf2-7ed0-4629-a6bf-feaf4ab7fed0	y	HSC	97	HSC-Y	0	424
calexp	HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z	ddeb90af-f966-4951-b01e-a886512e5c1f	y	HSC	62	HSC-Y	0	440

We have thus demonstrated that dataset discovery by a given set of DataUnits is enabled by the butler.

5.3.3.36 LVV-T2464 - Verify multiple simultaneous sky definitions

Version **1**. Open *LVV-T2464* test case in Jira.

Verify that a collection is able to hold Datasets corresponding to different sky tilings simultaneously

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Make empty repo.

Expected Result

Actual Result

Step 2 Step Execution Status: **Pass**

Description

Run ‘butler register-skymap‘.

Expected Result

Actual Result

Step 3 Step Execution Status: **Pass**

Description

Run ‘butler register-skymap‘ a second time

Expected Result

Actual Result

We will start with an existing repository that has more than one existing skymap. In particular, we will select a repo with recent reprocessing of the RC2 dataset.

Step 4 Step Execution Status: **Pass**

Description

Verify that mappings to both tile definitions are valid

Expected Result

Actual Result

On lsst-devl at NCSA, with the science pipelines set up, opened python and typed the following:

```
import lsst.daf.butler as dafButler
repo = "/repo/main"
butler = dafButler.Butler(repo, collections=["HSC/runs/RC2/w_2022_12/DM-34125"])
registry = butler.registry
for d in registry.queryDimensionRecords("skymap"):
   print(d)
```

This results in the following screen output:

```
skymap:
  name: 'hsc_rings_v1'
  hash: b'\xe2\x9f\xe9\xf1\x00\x8e5\x9f6\xa3g~}i\xccC\x93v\xd1\xe6'
  tract_max: 18938
  patch_nx_max: 9
  patch_ny_max: 9
skymap:
  name: 'hsc_rings_cells_v1'
  hash: b'\xde\x85\x13\xb0q\x11\x1e\x81a\x9e\\\x06\x1f\x02mA\xf7h\xe6\xd4'
  tract_max: 18938
  patch_nx_max: 11
  patch_ny_max: 11
```

This demonstrates that this single collection ("HSC/runs/RC2/w_2022_12/DM-34125") contains two sky maps with different numbers of patches (i.e., with different values of patch_nx_max and patch_ny_max).

5.3.3.37 LVV-T2465 - Verify pipeline execution in multiple contexts

Version **1**. Open *LVV-T2465* test case in Jira.

Verify that the design allows a given Pipeline specification to be used in both development and production contexts.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

This is a fundamental part of the design of PipelineTask.

Expected Result

Actual Result

5.3.3.38 LVV-T2461 - Verify Collection Layering: Science Platform

Version **1**. Open *LVV-T2461* test case in Jira.

Verify that collections created in the Science Platform are usable as inputs for processing initiated in the Science Platform

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Run part of DRP pipeline in RSP.

Expected Result

Actual Result

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

Description

Run a later part of DRP pipeline in RSP

Expected Result

Actual Result

5.3.3.39 LVV-T2463 - Verify enabling of different execution environments

Version **1**. Open *LVV-T2463* test case in Jira.

Verify that the supervisory framework supports the creation of multiple specializations for different execution environments.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Satisfied by BPS plugin system; we have plugins for many workflow systems already.

Expected Result

Actual Result

5.3.3.40 LVV-T2462 - Verify QuantumGraph algorithm

Version **1**. Open *LVV-T2462* test case in Jira.

Verify QuantumGraph algorithm common to all execution environments. Verify that the supervisory framework provides a common implementation of the logic required for interpretation of the Pipeline steps and their data groupings (and thus the possible parallelization); i.e., that the QuantumGraph generation algorithm can be common to all execution environments.

Preconditions:

Execution status: **Pass**

Final comment:
Working on lsst-devl machines in a cloned ‘pipe_base‘ repository at /project/jcarlin/SVV/gen3_middleware_ac

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute the test_graphBuilder.py and test_quantumGraph.py unit tests in the pipe_base package.

Expected Result

Successful execution of the unit tests.

Actual Result

First execute the unit test of a simple graph builder:

pytest -s -vv –no-header –cache-clear tests/test_graphBuilder.py

Result:

tests/test_graphBuilder.py::FLAKE8 PASSED
tests/test_graphBuilder.py::GraphBuilderTestCase::testAddInstrumentMismatch PASSED
tests/test_graphBuilder.py::GraphBuilderTestCase::testDefault PASSED

Now execute the unit test that more thoroughly tests quantum graph generation and usage:
pytest -s -vv –no-header –cache-clear tests/test_quantumGraph.py | tee test_QG_log.txt

tests/test_quantumGraph.py::FLAKE8 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testAllDatasetTypes PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testContains PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineAnsestorsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineConnectionsOfQuantum PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineOutputsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindCycle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindQuantaWIthDSType PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByLabel PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByName PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithInput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithOutput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetNodesForTask PASSED

tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantaForTask PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantumNodeByNodeId PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testInputQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testLength PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testOutputtQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testPickle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoad PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUri PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUriS3 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSubset PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSubsetToConnected PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskWithDSType PASSED
tests/test_quantumGraph.py::MyMemoryTestCase::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/min
py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py PASSED

All passed. This demonstrates that functional code is in place for generating quantum graphs.

5.3.3.41 LVV-T2460 - Verify generating a DAG

Version **1**. Open *LVV-T2460* test case in Jira.

Verify that the supervisory framework supports the "Pre-flight" phase of execution of a Pipeline on a specified set of inputs and/or desired outputs, resulting in a Directed Acyclic Graph (DAG) for the processing, with the nodes in the DAG being the units of work to be executed.

Preconditions:

Execution status: **Pass**

Final comment:
Working on lsst-devl machines in a cloned 'pipe_base' repository at /project/jcarlin/SVV/gen3_middleware_ac

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Satisfied by existence of QuantumGraph generation code.

Expected Result

Actual Result

First execute the unit test of a simple graph builder:

pytest -s -vv –no-header –cache-clear tests/test_graphBuilder.py

Result:

tests/test_graphBuilder.py::FLAKE8 PASSED
tests/test_graphBuilder.py::GraphBuilderTestCase::testAddInstrumentMismatch PASSED
tests/test_graphBuilder.py::GraphBuilderTestCase::testDefault PASSED

Now execute the unit test that more thoroughly tests quantum graph generation and usage:
pytest -s -vv –no-header –cache-clear tests/test_quantumGraph.py | tee test_QG_log.txt

tests/test_quantumGraph.py::FLAKE8 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testAllDatasetTypes PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testContains PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineAnsestorsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineConnectionsOfQuantum PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineOutputsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindCycle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindQuantaWIthDSType PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByLabel PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByName PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithInput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithOutput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetNodesForTask PASSED

tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantaForTask PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantumNodeByNodeId PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testInputQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testLength PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testOutputtQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testPickle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoad PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUri PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUriS3 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSubset PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSubsetToConnected PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskWithDSType PASSED
tests/test_quantumGraph.py::MyMemoryTestCase::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/min
py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py PASSED

All passed. This demonstrates that functional code is in place for DAG generation.

5.3.3.42 LVV-T2457 - Verify butler instantiation

Version **1**. Open *LVV-T2457* test case in Jira.

Verify that the supervisory framework creates and supplies the Butler required to support the I/O to be performed in the "Run" phase, for each unit of work.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.43 LVV-T2456 - Verify execution logging

Version **1**. Open *LVV-T2456* test case in Jira.

Verify that standard logging is enabled for the pre-flight and run processes of pipelines.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute unit tests in https://github.com/lsst/pipe_base/; in particular, test_logging.py, test_task.py, and test_pipelineTask.py.

Expected Result

Unit tests pass.

Actual Result

On lsst-devl machines at NCSA, working in a cloned ‘pipe_base‘ repository at /project/jcarlin/SVV/gen3_middleware_acceptance_testi

‘pytest -s -vv –no-header –cache-clear tests/test_logging.py‘

tests/test_logging.py::TestLogging::testLogCommands PASSED
tests/test_logging.py::TestLogging::testLogLevels PASSED

‘pytest -s -vv –no-header –cache-clear tests/test_task.py‘

tests/test_task.py::TaskTestCase::testBasics PASSED
tests/test_task.py::TaskTestCase::testEmptyMetadata PASSED
tests/test_task.py::TaskTestCase::testFail PASSED
tests/test_task.py::TaskTestCase::testGetFullMetadata PASSED
tests/test_task.py::TaskTestCase::testLog PASSED
tests/test_task.py::TaskTestCase::testNames PASSED
tests/test_task.py::TaskTestCase::testReplace PASSED
tests/test_task.py::TaskTestCase::testTimeMethod PASSED

‘pytest -s -vv –no-header –cache-clear tests/test_pipelineTask.py‘

tests/test_pipelineTask.py::PipelineTaskTestCase::testChain2 PASSED
tests/test_pipelineTask.py::PipelineTaskTestCase::testRunQuantum PASSED
tests/test_pipelineTask.py::MyMemoryTestCase::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/minico
py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py

The first of these demonstrates the logging and its different levels, and the other two unit tests exercise the logging explicitly in the tests. All have passed, so we have demonstrated logging of execution. (Note that pipetasks are called during both pre-flight and run processes, so these unit tests are sufficient for both cases.)

5.3.3.44 LVV-T2455 - Verify pipeline interface available as Python API

Version **1**. Open *LVV-T2455* test case in Jira.

Verify that the Pipeline specification interface is available as a Python API.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.45 LVV-T2454 - Verify pre-execution config overrides

Version **1**. Open *LVV-T2454* test case in Jira.

Verify that the middleware enables programmatic overrides to the configurations specified for a Pipeline, and that the overrides can be captured for purposes of provenance recording.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Expected Result

Actual Result

5.3.3.46 LVV-T2458 - Verify serialization of pre-flight results

Version **1**. Open *LVV-T2458* test case in Jira.

Verify that the supervisory framework provides a serialization form for the results of the "Pre-flight" phase, so that they can be computed in one process and executed under the control of one or more others.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Satisfied by QuantumGraph being serializable. Demonstrate using unit tests in https://github.com/lsst/pipe_base/tests/test_quantu

Expected Result

Unit test passes.

Actual Result

On lsst-devl machines at NCSA, working in a cloned 'pipe_base' repository at /project/jcarlin/SVV/gen3_middleware_acceptance_testi

Execute the unit tests that thoroughly tests quantum graph generation and usage:
pytest -s -vv –no-header –cache-clear tests/test_quantumGraph.py | tee test_QG_log.txt

tests/test_quantumGraph.py::FLAKE8 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testAllDatasetTypes PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testContains PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineAnsestorsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineConnectionsOfQuantum PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testDetermineOutputsOfQuantumNode PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindCycle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindQuantaWIthDSType PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByLabel PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTaskDefByName PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithInput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testFindTasksWithOutput PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetNodesForTask PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantaForTask PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGetQuantumNodeByNodeId PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testInputQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testLength PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testOutputtQuanta PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testPickle PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoad PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUri PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSaveLoadUriS3 PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testSubset PASSED

tests/test_quantumGraph.py::QuantumGraphTestCase::testSubsetToConnected PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskGraph PASSED
tests/test_quantumGraph.py::QuantumGraphTestCase::testTaskWithDSType PASSED
tests/test_quantumGraph.py::MyMemoryTestCase::testFileDescriptorLeaks <- ../../../../../software/lsstsw/stack_20220215/stack/mir
py38_4.9.2-2.0.0/Linux64/utils/g617c0b0dc2+9633a190c8/python/lsst/utils/tests.py PASSED

All passed. These tests (particularly the ones with "SaveLoad" in their names)demonstrate that the quantum graphs can be serialized and read in to initiate execution.

5.3.3.47 LVV-T2451 - Verify ability to append to an existing repository

Version **1**. Open *LVV-T2451* test case in Jira.

Verify that it is possible to add Datasets to a pre-existing Collection via additional processing.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute unit tests in `https://github.com/lsst/daf_butler/blob/main/tests/test_butler.py` (in particular, ButlerPutGetTests demonstrate creating a collection, then adding datasets to it.

Expected Result

Unit test passes

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_butler.py

Results:
tests/test_butler.py::PosixDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testBasicPutGet PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerMakeRepoOutfileTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetVirtual PASSED

All tests of the butler's "Put" and "Get" functionality passed. These tests first create a "run" collection, then append datasets to that collection, and thus demonstrate the required functionality.

5.3.3.48 LVV-T2453 - Verify creation of DatasetRef upon butler.put

Version **1**. Open *LVV-T2453* test case in Jira.

Verify that upon writing a dataset, a DatasetRef is created to enable getting the dataset in the future.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description
Execute ButlerPutGetTests in `https://github.com/lsst/daf_butler/blob/main/tests/test_butler.py`

Expected Result
Unit test passes

Actual Result
On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_butler.py

Results:
tests/test_butler.py::PosixDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetConcrete PASSED

tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::InMemoryDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ChainedDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testBasicPutGet PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::ButlerMakeRepoOutfileTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileDirTestCase::testPutGet PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testConfigExistence PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::ButlerMakeRepoOutfileUriTestCase::testPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::S3DatastoreButlerTestCase::testCompositePutGetVirtual PASSED

All tests of the butler's "Put" and "Get" functionality passed.

5.3.3.49 LVV-T2449 - Verify middleware writer configurability

Version **1**. Open *LVV-T2449* test case in Jira.

Verify that the data output system supports configuration of individual writer behavior.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description

Execute the unit test at `https://github.com/lsst/daf_butler/blob/main/tests/test_config.py`, which tests the writer configuration.

Expected Result

Unit test passes.

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_config.py

Results:

tests/test_config.py::FLAKE8 PASSED
tests/test_config.py::ConfigTestCase::testBadConfig PASSED
tests/test_config.py::ConfigTestCase::testBasics PASSED
tests/test_config.py::ConfigTestCase::testDict PASSED
tests/test_config.py::ConfigTestCase::testEscape PASSED
tests/test_config.py::ConfigTestCase::testHierarchy PASSED
tests/test_config.py::ConfigTestCase::testMerge PASSED
tests/test_config.py::ConfigTestCase::testOperators PASSED
tests/test_config.py::ConfigTestCase::testSerializedString PASSED
tests/test_config.py::ConfigTestCase::testSplitting PASSED
tests/test_config.py::ConfigTestCase::testUpdate PASSED
tests/test_config.py::ConfigSubsetTestCase::testAbsPath PASSED
tests/test_config.py::ConfigSubsetTestCase::testClassDerived PASSED
tests/test_config.py::ConfigSubsetTestCase::testDefaults PASSED
tests/test_config.py::ConfigSubsetTestCase::testEmpty PASSED
tests/test_config.py::ConfigSubsetTestCase::testExternalHierarchy PASSED

tests/test_config.py::ConfigSubsetTestCase::testExternalOverride PASSED
tests/test_config.py::ConfigSubsetTestCase::testInclude PASSED
tests/test_config.py::ConfigSubsetTestCase::testIncludeConfigs PASSED
tests/test_config.py::ConfigSubsetTestCase::testNoDefaults PASSED
tests/test_config.py::ConfigSubsetTestCase::testPathlib PASSED
tests/test_config.py::ConfigSubsetTestCase::testResource PASSED
tests/test_config.py::ConfigSubsetTestCase::testSearchPaths PASSED
tests/test_config.py::ConfigSubsetTestCase::testStringInclude PASSED
tests/test_config.py::FileWriteConfigTestCase::testDump PASSED

This confirms that the writer behavior can be configured.

5.3.3.50 LVV-T2452 - Verify specification of output locations

Version **1**. Open *LVV-T2452* test case in Jira.

Verify that the middleware enables configuration of the output location for a POSIX file system.

Preconditions:

Execution status: **Pass**

Final comment:
Working with a cloned 'daf_butler' repository at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/d
on the lsst-devl machines.

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description
Execute the PosixDatastoreButlerTestCase in `https://github.com/lsst/daf_butler/blob/main/tests/test_butler.py`

Expected Result

Unit test passes

Actual Result
Executed the unit test via: "pytest -s -vv –no-header tests/test_butler.py"

Results:

tests/test_butler.py::PosixDatastoreButlerTestCase::testBasicPutGet PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testButlerRewriteDataId PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetConcrete PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testCompositePutGetVirtual PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testConstructor PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testDeferredCollectionPassing PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testExportTransferCopy PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testGetDatasetTypes PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testImportExport Root: file:///project/jcarlin/SVV/gen3_middleware_acceptanc
PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testImportExportVirtualComposite Root: file:///project/jcarlin/SVV/gen3_middl
XFAIL
tests/test_butler.py::PosixDatastoreButlerTestCase::testIngest PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testMakeRepo PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPathConstructor PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPickle PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPruneCollections PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPruneDatasets PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPutTemplates PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPytypeCoercion PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testPytypePutCoercion PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testRemoveRuns PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testStringification PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testTransaction PASSED

All of the tests in PosixDatastoreButlerTestCase have passed.

5.3.3.51 LVV-T2450 - Verify writing dataset to multiple repositories

Version **1**. Open *LVV-T2450* test case in Jira.

Verify that the middleware enables writing of a single dataset to multiple repositories, with a different output format used for each repository.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Not regularly exercised in production, but functionality is tested in daf_butler unit tests. — WHICH UNIT TESTS?

Expected Result

Unit test passes

Actual Result

5.3.3.52 LVV-T2447 - Verify DataRepository layering: Data Release and Science Platform

Version **1**. Open *LVV-T2447* test case in Jira.

Verify that a Data Release is usable as the inputs for processing initiated in the Science Platform.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description

Reuse test scirpt for DMS-MWBT-REQ-0012

Expected Result

Actual Result

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

Description

Run a DRP pipeline subset in RSP, using DP0.x collections as inputs.

Expected Result

Actual Result

5.3.3.53 LVV-T2446 - Verify registries of collections

Version **1**. Open *LVV-T2446* test case in Jira.

Verify that there is a mechanism for registering Collections as they are created

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute 'butler query-collections' to show that collections are registered and searchable.

Expected Result

Actual Result

On lsst-devl machines at NCSA, set up the Science Pipelines, then run:

butler query-collections /repo/main *DM-341*

The glob "*DM-341*" should locate any collection with that string in its name or path. (This is an arbitrarily chosen string that should find collections related to Jira tickets with ticket numbers DM-341??.)

Here is a portion of the output from this query:

Name Type
——————————————————————- ———–
HSC/runs/RC2/w_2022_12/DM-34125 CHAINED
HSC/runs/RC2/w_2022_12/DM-34125/20220324T205113Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T213046Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T211319Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220323T173939Z RUN

HSC/runs/RC2/w_2022_12/DM-34125/20220321T153517Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z RUN
HSC/raw/RC2/9615 TAGGED
HSC/raw/RC2/9697 TAGGED
HSC/raw/RC2/9813 TAGGED
HSC/calib/DM-32378 CALIBRATION
HSC/calib/gen2/20180117 CALIBRATION
HSC/calib/DM-28636 CALIBRATION
HSC/calib/gen2/20180117/unbounded RUN
HSC/calib/DM-28636/unbounded RUN
HSC/masks/s18a RUN
HSC/fgcmcal/lut/RC2/DM-28636 RUN
refcats/DM-28636 RUN
skymaps RUN
refcats/DM-33444 RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220321T153517Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220323T173939Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220324T205113Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T211319Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T213046Z RUN
u/yusra/RC2/w_2022_12/DM-34125 CHAINED
u/yusra/RC2/w_2022_12/DM-34125/20220404T154651Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T213046Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220325T211319Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220324T205113Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220323T173939Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220321T222013Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220321T153517Z RUN
HSC/runs/RC2/w_2022_12/DM-34125/20220319T213338Z RUN
HSC/raw/RC2/9615 TAGGED
HSC/raw/RC2/9697 TAGGED
HSC/raw/RC2/9813 TAGGED
HSC/calib/DM-32378 CALIBRATION
HSC/calib/gen2/20180117 CALIBRATION
HSC/calib/DM-28636 CALIBRATION

We have thus demonstrated that Collections are registered in butler databases.

5.3.3.54 LVV-T2444 - Verify dataset garbage collection

Version **1**. Open *LVV-T2444* test case in Jira.

Verify that when a DataRepository is removed, the Datasets it references are removed if and only if they are not also referenced by one or more additional DataRepositories that have been explicitly identified.

Note that the requirement text assumed a slightly different collections model from what we have. Instead of "reference counting" datasets, we have RUN collections that own datasets and TAGGED collections that don't, but we still guard against improper deletions as the requirement demands.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Make example repo, one each, POSIX and S3

Expected Result

Actual Result

Step 2 Step Execution Status: **Pass**

Description

Create TAGGED collection and add some datasets to it.

Expected Result

Actual Result

Step 3 Step Execution Status: **Pass**

Description

Try to delete the RUN collection - shouldn't be possible because of references in TAGGED collection.

Expected Result

Actual Result

Step 4 Step Execution Status: **Pass**

Description

Try to delete the TAGGED collection - should work, without deleting the datasets.

Expected Result

Actual Result

All of the steps suggested in this test script are executed in the test_cliCmdPruneCollection.py unit test from daf_butler. Execute this:

pytest -s -vv –no-header –cache-clear tests/test_cliCmdPruneCollection.py

tests/test_cliCmdPruneCollection.py::PruneCollectionsTest::testPruneCollections PASSED
tests/test_cliCmdPruneCollection.py::PruneCollectionExecutionTest::testPruneRun PASSED
tests/test_cliCmdPruneCollection.py::PruneCollectionExecutionTest::testPruneTagged PASSED

The unit test includes verification that a TAGGED collection can be removed, and that RUN collections cannot be removed without purging and unstoring the datasets.

5.3.3.55 LVV-T2442 - Verify dataset deletion

Version **1**. Open *LVV-T2442* test case in Jira.

Verify that a Dataset is deletable from a DataRepository by an authorized person.

Preconditions:

Execution status: **Initial Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Initial Pass**

Description

Make example repo, one each, POSIX and S3

Expected Result

Actual Result

Step 2	Step Execution Status: **Pass**

Description

Run 'butler prune-datasets'

Expected Result

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_cliCmdPruneDatasets.py

The results show that the deletion of datasets was successful:

tests/test_cliCmdPruneDatasets.py::FLAKE8 PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_defaults_doContinue PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_defaults_doNotContinue PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_disassociateImpliedArgs PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_disassociateImpliedArgsWithCollections PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_dryRun_disassociate PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_dryRun_unstore PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_dryRun_unstoreAndDisassociate PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_noCollections PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_noConfirm PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_noDatasets PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_purgeImpliedArgs PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_purgeImpliedArgsWithCollections PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_purgeOnNonRunCollection PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_purgeWithDisassociate PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_quiet PASSED
tests/test_cliCmdPruneDatasets.py::PruneDatasetsTestCase::test_quietWithDryRun PASSED

Step 3	Step Execution Status: **Pass**

Description

Verify that the datasets are deleted

Expected Result

Actual Result
The unit test includes steps that confirm the deletion of the datasets.

5.3.3.56 LVV-T2443 - Verify repository removal

Version **1**. Open *LVV-T2443* test case in Jira.

Verify that an authorized user can remove a DataRepository from any storage environment. Verification on **all** environments is not possible. We will verify POSIX and S3 environments, which we believe is in the spirit of the requirement and covers our core needs.

Preconditions:

Execution status: **Initial Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Initial Pass**

Description
Make example repo, one each, POSIX and S3

Expected Result

Actual Result

Tested on a POSIX datastore at NCSA (lsst-devl machines).

Step 2 Step Execution Status: **Pass**

Description

Run 'butler prune-collection'

Expected Result

Actual Result

On lsst-devl machines at NCSA, in a cloned 'daf_butler' repository (at /project/jcarlin/SVV/gen3_middleware_acceptance_testing/daf_
execute:

pytest -s -vv –no-header –cache-clear tests/test_cliCmdPruneCollection.py

The results show that the pruning of collections is successful:

tests/test_cliCmdPruneCollection.py::FLAKE8 PASSED
tests/test_cliCmdPruneCollection.py::PruneCollectionsTest::testPruneCollections PASSED
tests/test_cliCmdPruneCollection.py::PruneCollectionExecutionTest::testPruneRun PASSED
tests/test_cliCmdPruneCollection.py::PruneCollectionExecutionTest::testPruneTagged PASSED

Step 3 Step Execution Status: **Pass**

Description

Verify that the repository is removed

Expected Result

Actual Result

The unit test includes checks that the collection has been removed.

5.3.3.57 LVV-T2441 - Verify repository version migration

Version **1**. Open *LVV-T2441* test case in Jira.

Verify that the Data Input/Output system can perform persistent migrations of a DataRepository to bring the Data Model of that DataRepository up to parity with the Data Model expected by the current Data Input/Output System interfaces.

Preconditions:

Execution status: **Not Executed**

Final comment:

Detailed steps results:

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

Description
Make example repo with one schema configuration (e.g. autoincrement dataset IDs).

Expected Result

Actual Result

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

Description
Run existing migration script that upgrades the repository to the new one.

Expected Result

Actual Result

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

Description
Verify that the features are migrated correctly

Expected Result

Actual Result

5.3.3.58 LVV-T2440 - Verify versioning of DataRepositories

Version **1**. Open *LVV-T2440* test case in Jira.

Verify that the Data Input/Output system can describe the version of a DataRepository

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1	Step Execution Status: **Pass**

Description
Print attributes table via Butler APIs, against literally any repo.

Expected Result

Actual Result
Open the database for a repository on NCSA machines:

sqlite3 /project/jcarlin/repos/rc2_subset/SMALL_HSC/gen3.sqlite3

Output the butler_attributes table to a file:
sqlite> .output attributes.txt
sqlite> SELECT * from butler_attributes;

The first few lines of this output file are:
config:registry.managers.attributes|lsst.daf.butler.registry.attributes.DefaultButlerAttributeManager
config:registry.managers.dimensions|lsst.daf.butler.registry.dimensions.static.StaticDimensionRecordStorageManager
config:registry.managers.collections|lsst.daf.butler.registry.collections.synthIntKey.SynthIntKeyCollectionManager
config:registry.managers.datasets|lsst.daf.butler.registry.datasets.byDimensions._manager.ByDimensionsDatasetRecordStorageM
config:registry.managers.opaque|lsst.daf.butler.registry.opaque.ByNameOpaqueTableStorageManager
config:registry.managers.datastores|lsst.daf.butler.registry.bridge.monolithic.MonolithicDatastoreRegistryBridgeManager
version:lsst.daf.butler.registry.attributes.DefaultButlerAttributeManager|1.0.0
schema_digest:lsst.daf.butler.registry.attributes.DefaultButlerAttributeManager|664d6a56d87b5ac890308a91a06cd145
version:lsst.daf.butler.registry.dimensions.static.StaticDimensionRecordStorageManager|6.0.0
schema_digest:lsst.daf.butler.registry.dimensions.static.StaticDimensionRecordStorageManager|83022175a1fbb71edd4f5243a17
version:lsst.daf.butler.registry.collections.synthIntKey.SynthIntKeyCollectionManager|2.0.0
schema_digest:lsst.daf.butler.registry.collections.synthIntKey.SynthIntKeyCollectionManager|1d45208fb4ad1b51bed29321deb787
version:lsst.daf.butler.registry.datasets.byDimensions._manager.ByDimensionsDatasetRecordStorageManagerUUID|1.0.0
schema_digest:lsst.daf.butler.registry.datasets.byDimensions._manager.ByDimensionsDatasetRecordStorageManagerUUID|338a
version:lsst.daf.butler.registry.opaque.ByNameOpaqueTableStorageManager|0.2.0
schema_digest:lsst.daf.butler.registry.opaque.ByNameOpaqueTableStorageManager|79a657af5cf15550e6d1f455ad4dd8c2
version:lsst.daf.butler.registry.bridge.monolithic.MonolithicDatastoreRegistryBridgeManager|0.2.0
schema_digest:lsst.daf.butler.registry.bridge.monolithic.MonolithicDatastoreRegistryBridgeManager|3558b84d12fa04082ffd6935

This demonstrates that the versions are recorded in the butler_attributes table of a dataRepository.

5.3.3.59 LVV-T2439 - Verify relocatability of DataRepositories

Version **1**. Open *LVV-T2439* test case in Jira.

Verify that DataRepositories can be relocated between various storage contexts.

Preconditions:

Execution status: **Pass**

Final comment:

Detailed steps results:

Step 1 Step Execution Status: **Pass**

Description

Execute unit tests in https://github.com/lsst/daf_butler/tests/test_butler.py, which thoroughly exercise all aspects of transferring data repositories.

Expected Result

Actual Result

In a cloned version of the daf_butler repository, execute:

pytest -s -vv –no-header –cache-clear tests/test_butler.py | tee test_butler_log.txt

Among the many unit tests executed by this command, the following are the relevant tasks that demonstrate relocatability of dataRepositories:

tests/test_butler.py::PosixDatastoreButlerTestCase::testExportTransferCopy PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testGetDatasetTypes PASSED
tests/test_butler.py::PosixDatastoreButlerTestCase::testImportExport Root: file:///project/jcarlin/SVV/gen3_middleware_acceptanc
PASSED

tests/test_butler.py::ButlerExplicitRootTestCase::testExportTransferCopy PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testFileLocations PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testGetDatasetTypes PASSED
tests/test_butler.py::ButlerExplicitRootTestCase::testImportExport Root: file:///project/jcarlin/SVV/gen3_middleware_acceptance_te
PASSED

tests/test_butler.py::PosixDatastoreTransfers::testTransferIntToInt PASSED
tests/test_butler.py::PosixDatastoreTransfers::testTransferIntToUuid PASSED
tests/test_butler.py::PosixDatastoreTransfers::testTransferMissing PASSED
tests/test_butler.py::PosixDatastoreTransfers::testTransferMissingDisassembly PASSED
tests/test_butler.py::PosixDatastoreTransfers::testTransferUuidToUuid PASSED

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
1D	One-dimensional
2D	Two-dimensional
ADC	atmospheric dispersion corrector
API	Application Programming Interface
BOT	Bench for Optical Testing
BPS	Batch Production Service
CCB	Change Control Board
CI	Continuous Integration
CPP	Calibration Production Processing
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.
DBB	Data Backbone
DECam	Dark Energy Camera
DM	Data Management
DMS	Data Management Subsystem
DMTN	DM Technical Note
DOM	Document Object Model
DP0	Data Preview 0
DRP	Data Release Production
FITS	Flexible Image Transport System
HSC	Hyper Suprime-Cam
IRSA	Infrared Science Archive
LATISS	LSST Atmospheric Transmission Imager and Slitless Spectrograph
LDM	LSST Data Management (Document Handle)

LSE	LSST Systems Engineering (Document Handle)
LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
LVV	LSST Verification and Validation
NCSA	National Center for Supercomputing Applications
OODS	Observatory Operations Data Service
PDF	Portable Document Format
PMCS	Project Management Controls System
POSIX	Portable Operating System Interface
PSF	Point Spread Function
RA	Right Ascension
RSP	Rubin Science Platform
S3	(Amazon) Simple Storage Service
SLAC	SLAC National Accelerator Laboratory
URL	Universal Resource Locator
WCS	World Coordinate System
WISE	Wide-field Survey Explorer
arcsec	arcsecond second of arc (unit of angle)
bps	bit(s) per second
deg	degree; unit of angle