LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Data Management Middleware Requirements

Gregory Dubois-Felsmann, Tim Jenness, Jim Bosch, Michelle Gower, Simon Krughoff, Russell Owen, Pim Schellart, Brian van Klaveren

LDM-556

Latest Revision: 2017-12-04

Change Record

Version	Date	Description	Owner name
	2017-05-19	Initial version based on SuperTask working group recommendations.	G. Dubois-Felsmann
	2017-11-30	Add initial draft of Data Access (butler) requirements	T. Jenness

Document source location: MagicDraw SysML
Version from source repository: 62

Contents

Data Management Middleware Requirements

1 Data Access Abstraction Layer

This section describes the requirements relating to the data access abstraction layer (commonly referred to as ”the butler”) as derived from requirement DMS-REQ-0298. Where Use Cases are mentioned, they are defined in LDM-592. LDM-592 also includes a glossary.

1.1 Data Repositories

1.1.1 Data Repository Upload

ID: DMS-MWBT-REQ-0019 (Priority: 1a)

Specification: It shall be possible to explicitly transfer a Data Repository or subset thereof from external hardware to the Science Platform.

Discussion: This is effectively a weaker version of DMS-MWBT-REQ-0012 for these two storage contexts, permitting the system to require the user to perform an explicit upload instead of expecting a seamless connection. (UseCases: SCIVAL1, SCIVAL3, SCIVAL4)

1.1.2 Dataset Deletion

ID: DMS-MWBT-REQ-0004 (Priority: 1a)

Specification: A Dataset shall be deletable from a Data Repository by an authorized person.

Discussion: Authorization means whatever mechanism provided by storage system (e.g., POSIX permissions) (UseCases: LDF7, DRP30)

1.1.3 Dataset Garbage Collection

ID: DMS-MWBT-REQ-0006 (Priority: 1b)

Specification: When a Data Repository is removed, the Datasets it references shall be removed if and only if they are not also referenced by one or more additional Data Repositories that have been explicitly identified.

Discussion: The ”additional Data Repositories” may simply be those defined in the same database system used to define the Data Repository that is being removed. There is no expectation that all Data Repositories that reference to a Dataset can be tracked in general. (UseCases: DRP29)

1.1.4 DataUnit Update

ID: DMS-MWBT-REQ-0023 (Priority: 2)

Specification: It shall be possible to create a new Data Repository that contains DataUnits whose metadata and relationship values are defined by processing outputs in another Data Repository.

Discussion: This permits values like the WCS or PSF size associated with a visit to be improved with processing results. Note that this is not a requirement that existing Data Repositories permit their DataUnits to be updated in-place. (UseCases: DRP14)

1.1.5 LSST Data Ingest: calibration

ID: DMS-MWBT-REQ-0009 (Priority: 1a)

Specification: The Data Repository Creation System shall be able to ingest raw LSST calibration frames into a local DataRepository outside the archive center.

Discussion: (UseCases: DRP10)

1.1.6 LSST Data Ingest: science

ID: DMS-MWBT-REQ-0008 (Priority: 1a)

Specification: The Data Repository Creation System shall be able to ingest raw LSST science images into a local DataRepository outside the archive center.

Discussion: (UseCases: DRP10)

1.1.7 Multiple Cameras

ID: DMS-MWBT-REQ-0022 (Priority: 2)

Specification: A Data Repository shall be able to hold Datasets with Data Units corresponding to different cameras simultaneously.

Discussion: This would include simulated data. (UseCases: DRP11, COMM11)

1.1.8 Registries of Data Repositories

ID: DMS-MWBT-REQ-0024 (Priority: 1a)

Specification: There shall be a mechanism for registering Data Repositories as they are created.

Discussion: For example, if I have several Data Repositories in a single storage context, I shouldn't have to change how I refer to those repositories if the storage context changes. This also allows for discoverability in, e.g. the release registry. Registration should be automatic. (UseCases: SQR9)

1.1.9 Relocatability of Data Repositories

ID: DMS-MWBT-REQ-0001 (Priority: 1a)

Specification: Data Repositories shall be relocatable between various storage contexts.

Discussion: A commissioning scientist who adds value to a local repository will want to share that repository via the commissioning archive or VOSpace. (UseCases: COMM3)

1.1.10 Repository Merging

ID: DMS-MWBT-REQ-0007 (Priority: 1b)

Specification: It shall be possible to merge multiple Data Repositories into a single Data Repository, given a strategy to resolve conflicts between the input Data Repositories.

Discussion: For example, batch jobs submitted from a notebook might each write a small local Data Repository but the user of the notebook wants the outputs to appear in a single repository without knowing how many jobs were submitted. (UseCases: ARCH4)

1.1.11 Repository Removal

ID: DMS-MWBT-REQ-0005 (Priority: 1a)

Specification: It shall be possible for an authorized user to remove a Data Repository from any storage environment.

Discussion: Some Data Repositories (e.g. Data Releases) may not have any thusly authorized users. This would also involve removing it from registries. Removing the repository might require database table modifications. (UseCases: LDF3, DRP29)

1.1.12 Repository version migration

ID: DMS-MWBT-REQ-0003 (Priority: 2)

Specification: The Data Input/Output system shall be able to 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.

Discussion: This is a tool for creating a repository from a repository with an old version to use the current version. Silent in-place updates of a repository should not occur. (UseCases: DAX8)

1.1.13 Subsetting a DataRepository with data transfer

ID: DMS-MWBT-REQ-0011 (Priority: 1a)

Specification: It shall be possible to easily create a new DataRepository which contains a copy of a sub-section of an existing DataRepository, given a list of DataUnits and a list of DatasetTypes.

Discussion: This would transfer the files but not load them into Python objects, thus allowing for instance a processing run to be done without network connection. (UseCases: DRP16, DAX1, LDF103, SQR1)

1.1.14 Subsetting a DataRepository without data transfer

ID: DMS-MWBT-REQ-0010 (Priority: 1a)

Specification: It shall be possible to easily create a new DataRepository which is a view of a sub-section of an existing DataRepository, given a list of DataUnits and a list of DatasetTypes.

Discussion: That is, given a list of DataUnits and a list of DatasetTypes, create a new DataRepository with enough information to access any of the DatasetTypes for which the necessary Data Units exist for the input list of DataUnits. This does not involve copying datasets. (UseCases: SQR1, AP1dev)

1.1.15 Versioning of Data Repositories

ID: DMS-MWBT-REQ-0002 (Priority: 1b)

Specification: The Data Input/Output system shall be able to describe the version of a DataRepository.

Discussion: (UseCases: DAX8)

1.1.16 Data Repository Layering Requirements

1.1.16.1 Data Repository Layering

ID: DMS-MWBT-REQ-0012 (Priority: 1a)

Specification: A DataRepository (A) shall be usable as an input for processing in a context (B), with its contents appearing as part of the Data Repository used to hold the outputs of the processing, for certain combinations of (A) and (B).

Discussion: Generally speaking, smaller-scale processing runs initiated by users with fewer permissions should be able to build on larger-scale processing runs iniated by users with more permissions. This requirement probably cannot be satisfied efficiently by always copying the full original input data repository (A) to the final output repository (B); it almost certainly implies some kind of on-demand transfer or aliasing. (UseCases: DRP1, DRP2, DRP3, DRP7, DRP8, SCIVAL1, SCIVAL2, SCIVAL3, SCIVAL4)

1.1.16.2 Data Repository Layering: Data Release and external hardware

ID: DMS-MWBT-REQ-0014 (Priority: 1a)

Specification: A Data Release (A) shall be usable as the inputs for test/development processing on external hardware (B).

Discussion: (UseCases: DRP1, DRP2, DRP3, DRP7, DRP8, SCIVAL1, SCIVAL3)

1.1.16.3 Data Repository Layering: Data Release and Science Platform

ID: DMS-MWBT-REQ-0013 (Priority: 1a)

Specification: A Data Release (A) shall be usable as the inputs for processing initiated in the Science Platform (B).

Discussion: (UseCases: DRP1, DRP2, DRP3, SCIVAL1)

1.1.16.4 Data Repository Layering: Data Release Production

ID: DMS-MWBT-REQ-0015 (Priority: 1a)

Specification: Intermediate outputs of Data Release Production [test] processing (A) shall be usable as inputs for later Data Release Production [test] processing (B).

Discussion: These ”intermediates” are normal pipeline outputs that may not be included in a formal data release, and this requirement only applies when these have not been discarded or elided. (UseCases: DRP1, DRP2, DRP3, SCIVAL2)

1.1.16.5 Data Repository Layering: Data Release Production intermediates to external hardware

ID: DMS-MWBT-REQ-0016 (Priority: 1a)

Specification: Intermediate outputs of Data Release Production [test] processing (A) shall be usable as inputs for test/development processing on external hardware (B).

Discussion: (UseCases: DRP1, DRP2, DRP3, DRP7, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.1.16.6 Data Repository Layering: Science Platform

ID: DMS-MWBT-REQ-0017 (Priority: 1a)

Specification: Data Repositories created in the Science Platform (A) shall be usable as inputs for processing initiated in the Science Platform (B).

Discussion: (UseCases: DRP1, DRP2, DRP3, DRP7, DRP8, SCIVAL1, SCIVAL3, SCIVAL4)

1.1.16.7 Data Repository Layering: Science Platform to external hardware

ID: DMS-MWBT-REQ-0018 (Priority: 1a)

Specification: Data Repositories created in the Science Platform (A) shall be usable as inputs for test/development processing on external hardware (B).

Discussion: (UseCases: DRP1, DRP2, DRP3, DRP7, DRP8, SCIVAL1, SCIVAL3)

1.1.17 Sky Tiles

1.1.17.1 Sky Tile Definition

ID: DMS-MWBT-REQ-0020 (Priority: 1a)

Specification: It shall be possible to add a new tiling of the sky (defined in a configuration file or code object) to a DataRepository programmatically.

Discussion: This allows the tracts and patches it defines to be used to identify Datasets. (UseCases: DRP13, SCIVAL1)

1.1.17.2 Multiple simultaneous sky tile definitions

ID: DMS-MWBT-REQ-0021 (Priority: 2)

Specification: A Data Repository shall be able to hold Datasets corresponding to different sky tilings simultaneously.

Discussion: We may have code that maps images between differently-defind tiles, or uses different tilings for different purposes in the same pipeline. (UseCases: DRP13)

1.2 Data Discovery System

1.2.1 DataRepository metadata lookup

ID: DMS-MWBT-REQ-0091 (Priority: 2)

Specification: It shall be possible to use the Data Discovery system to obtain metadata corresponding to a Dataset without reading the file(s) used to store the Dataset, as long as the desired metadata entries are identified when the DatasetType is defined.

Discussion: If the system stores metadata, the batch processing service should be able to ask the DataRepository for the metadata of particular DatasetRefs. This can be useful for debugging a batch processing job. (UseCases: LDF1)

1.2.2 Dataset Overrides

ID: DMS-MWBT-REQ-0087 (Priority: 1b)

Specification: It shall be possible for an operator to configure the Data Discovery System to override certain Datasets with others before retrieval.

Discussion: This allows operators to override master calibration files for a particular Batch Processing run. It could be implemented by DataRepository chaining or by marking some Datasets as ”preferred”. See also related requirement DMS-MWST-REQ-0016. (UseCases: DRP15, LDF1)

1.2.3 DataUnit lookup: processing driven

ID: DMS-MWBT-REQ-0080 (Priority: 1a)

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

Discussion: Answering the question of what data could possibily be used to build a coadd associated with this DataRef? (UseCases: SQR1, SQR9)

1.2.4 Filter by config

ID: DMS-MWBT-REQ-0090 (Priority: 1b)

Specification: The Data Discovery System shall be able to filter search results based upon user-specified filters containing explicit Datasets to be removed from results.

Discussion: This could be a list of raw data files in a text file that should not be included in the processing but which are not yet globally flagged. (UseCases: LDF1, LDF101)

1.2.5 Filter by data quality

ID: DMS-MWBT-REQ-0089 (Priority: 1b)

Specification: The Data Discovery System shall be able to filter search results based on data quality assessments.

Discussion: For example, ask for raw data that has been flagged as bad to not be included in a coadd. (UseCases: LDF102, LDF1)

1.2.6 Filter by non-DatasetRef Database Entries

ID: DMS-MWBT-REQ-0088 (Priority: 1a)

Specification: The Data Discovery System shall be able to filter search results based upon specified filters that need non-DatasetRef database entries.

Discussion: This includes joins with LDF Operator specific tables not normally known to the Data Discovery System. These cuts could include, but will not be limited to, seeing, data quality flags, and airmass. (UseCases: LDF1, SQR2, COMM7, SQR1, SQR2, LDF102)

1.2.7 Introspection for DatasetExpressions

ID: DMS-MWBT-REQ-0092 (Priority: 3)

Specification: The Data Discovery System shall allow for a DatasetExpression to be constructed interactively using introspection on the DataRepository schema

Discussion: An example of this would be tab-complete of DatasetTypes, which will make it easier for operators and astronomers to construct a DatasetExpression for an unknown DatasetRepository. Note that this could be provided by separate tooling, as long as sufficient information is available to develop such tooling. (UseCases: DRP23)

1.2.8 Multiple chained input DataRepositories

ID: DMS-MWBT-REQ-0081 (Priority: 1a)

Specification: The Data Discovery System shall be able treat multiple input DataRepositories as a single coherent logical repository.

Discussion: This could be a local on disk repository and a remote repository, with the the Data Discovery system scanning each in turn. Each dataset read in will contain provenance describing the Data Repository it came from. (UseCaseS: COMM4, LDF104)

1.2.9 Multiple parallel input DataRepositories

ID: DMS-MWBT-REQ-0082 (Priority: 1a)

Specification: The Data Discovery System shall be able to locate Datasets from multiple input DataRepositories in order to retrieve the same logical Dataset from them all.

Discussion: This is to allow for comparison of the same data reduced with multiple different stacks. These need to be both local and remote and combinations of the two. It could also be different versions of a data release. This is not a requirement on whether this is two butlers or one, although some other requirements imply a single system that knows about all repositories in a particular context. (UseCases: SQR7, LDF104, SQR1.5)

1.2.10 Discovery Interface Consistency

1.2.10.1 Consistent Discovery Interface

ID: DMS-MWBT-REQ-0083 (Priority: 1a)

Specification: The Data Discovery System shall provide a consistent interface for obtaining a graph that represents the DataUnits and Datasets in a Data Repository that match user-specified criteria.

Discussion: This is an interface expected by SuperTask preflight, and we need to make it consistent in all contexts in which SuperTasks will be launched. The same interface may be used in (possibly interactive) analysis and validation work. Note that many common queries on DataRepository contents may result in simple graphs that can be iterated as flat lists. (UseCases: DRP1, DRP7, SCIVAL1, SCIVAL2, SCIVAL3, AP2, DRP27, COMM8, COMM10, COMM13)

1.2.10.2 Data Discovery for Data Release Production

ID: DMS-MWBT-REQ-0084 (Priority: 1a)

Specification: The Data Discovery System interface shall be usable when initiating processing for Data Release Production.

Discussion: The interface does not need to be available in the compute environment in which jobs run (just the environment in which they are launched). (UseCases: DRP1, DRP7, SCIVAL1, SCIVAL2, SCIVAL3, AP2, DRP27, COMM8, COMM10, COMM13)

1.2.10.3 Data Discovery for notebook batch processing

ID: DMS-MWBT-REQ-0086 (Priority: 1a)

Specification: The Data Discovery System interface shall be usable when initiating batch or local processing in the Science Platform.

Discussion: The interface does not need to be available in the compute environment in which jobs run (just the environment in which they are launched). (UseCases: DRP1, DRP7, SCIVAL1, SCIVAL2, SCIVAL3, AP2, DRP27, COMM8, COMM10, COMM13)

1.2.10.4 Data Discovery for test processing runs

ID: DMS-MWBT-REQ-0085 (Priority: 1a)

Specification: The Data Discovery System interface shall be usable when initiating processing runs initiated for test/development purposes (on LSST or personal hardware),

Discussion: The interface does not need to be available in the compute environment in which jobs run (just the environment in which they are launched). (UseCases: DRP1, DRP7, SCIVAL1, SCIVAL2, SCIVAL3, AP2, DRP27, COMM8, COMM10, COMM13)

1.3 Data Input/Output System

1.3.1 Dataset Storage Elision

ID: DMS-MWBT-REQ-0027 (Priority: 1b)

Specification: It shall be possible to configure the Data Input/Output System such that Datasets of specific types are simply held in memory instead of written to storage when the Data Output System is invoked, and simply retrieved from memory when requested via the Data Input System, as long as both operations happen within the same process.

Discussion: We want the Data Input/Output System to pass all information between SuperTasks, but for performance reasons we don't always want this to involve I/O. (UseCases: DRP4)

1.3.2 Dump current configuration

ID: DMS-MWBT-REQ-0026 (Priority: 1b)

Specification: A mechanism shall be available for dumping the active configuration of the Data I/O system in human-readable form.

Discussion: Especially important if configuration comes from multiple sources and is required to be validated before submitting processing jobs. (UseCases: LDF1, LDF3)

1.3.3 Format pluggability

ID: DMS-MWBT-REQ-0025 (Priority: 1a)

Specification: It shall be 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.

Discussion: For example, raw data could be configured to be read in FITS format, but a calexp could be configured to be written in HDF5 format. Additionally, a calexp could be read in from HDF5 but appear in Python as an Astropy object rather than a AFW object; or a table could be persisted as a plot in PNG format. (UseCases: ARCH1, SQR4, SQR6)

1.3.4 I/O using cloud storage

ID: DMS-MWBT-REQ-0031 (Priority: 1a)

Specification: The Data Input/Output System shall be able to utilize cloud-based storage engines.

Discussion: For example Amazon's S3. In the case of CI, multiple CI jobs hosted in containers will all need parts of the same data, but it's too large to host locally. S3 allows each job to pull local only the data necessary for processing in that job. (UseCases: CI3)

1.3.5 I/O using distributed file system

ID: DMS-MWBT-REQ-0030 (Priority: 1a)

Specification: The Data Input/Output System shall be able to read/write from/to distributed file systems.

Discussion: The commissioning cluster will provide resources both for ad hoc and batch style processing. Both will likely utilize the same storage context. (UseCases: COMM1)

1.3.6 Science Platform VOSpace

ID: DMS-MWBT-REQ-0029 (Priority: 1a)

Specification: The Data Input/Output System interface shall provide access to the shared VOSpace file system from Jupyter notebooks running on the Science Platform

Discussion: (UseCases: DRP2, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.3.7 VOSpace

ID: DMS-MWBT-REQ-0028 (Priority: 1a)

Specification: It shall be possible to implement a Data Input/Output System that can operate on a repository located in a VOSpace.

Discussion: This means it should only pull data when it's needed. It shouldn't simply stage the entire repository to local disk as that is very inefficient. (UseCases: SQR14,LDF103)

1.4 Data Input

1.4.1 Aliases to Selections on Catalogs

ID: DMS-MWBT-REQ-0056 (Priority: 1b)

Specification: The Data Input System shall support aliases to selections on catalog data so that different Tasks may refer to the same subset of a catalog Dataset by name.

Discussion: This puts the onus on the database to store the aliases. It would be nice to see an implementation of a plugin that is able to set up the views from configuration if they don't already exist. It would be nice to support parameterized views, but this is not strictly necessary. (UseCases: SQR5)

1.4.2 Enabling SuperTasks to execute

ID: DMS-MWBT-REQ-0053 (Priority: 1a)

Specification: It shall be possible for the Data Input System to construct a InMemoryDataset from a set of files stored locally on disk (without a remote database connection).

Discussion: For example, the batch processing system will have retrieved a valid set of files from the Data Backbone and copied them to a local disk. The Data Input System will be reading from that local disk. (UseCases: LDF1, LDF3)

1.4.3 External Data Ingest

ID: DMS-MWBT-REQ-0046 (Priority: 1a)

Specification: The Data Input System shall be able to store non-LSST Datasets in a DataRepository

Discussion: (UseCases: DRP11)

1.4.4 External Data Ingest and Serve

ID: DMS-MWBT-REQ-0047 (Priority: 1a)

Specification: The Data Input System shall be able to load non-LSST Datasets from a DataRepository and serve them in the same manner as LSST Datasets (provided enough information is present in them)

Discussion: (UseCases: DRP11)

1.4.5 Failure on missing input file

ID: DMS-MWBT-REQ-0054 (Priority: 1a)

Specification: It shall be possible via configuration to require the Data Input System to fail if an expected file is not found at the specified location.

Discussion: During Batch Processing Service compute jobs, if the input file isn't on the local disk, the desired behavior is failure (as opposed to trying to get the file from the Data Backbone) (UseCases: LDF1)

1.4.6 Input Staging

ID: DMS-MWBT-REQ-0052 (Priority: 2)

Specification: The Data Input System shall be able to transfer Datasets selected by the SuperTask pre-flight stage from persistent storage to compute nodes for batch processing.

Discussion: (Note: Not currently planned to be used by the Operations Batch Processing Service) (UseCases: DRP5, DRP6, SCIVAL1, SCIVAL2)

1.4.7 Item from Composite Datasets

ID: DMS-MWBT-REQ-0034 (Priority: 1a)

Specification: It shall be possible to load into memory an item from a Composite Dataset without loading the full Dataset.

Discussion: An example of this is reading just the PSF object from an Exposure. (UseCases:

DRP18, DAX3, SQR15, COMM3)

1.4.8 Local caching of remote resources

ID: DMS-MWBT-REQ-0058 (Priority: 1a)

Specification: It shall be possible to configure the Data Input System to cache a local version of a Dataset that has been retrieved from a remote DataRepository.

Discussion: For example, when running an LSP Notebook, the first time it runs the data will be retrieved from, say, the object store or VO web service, but the second time the notebook is run it will use a cached version for increased efficiency. This functionality only needs to be implemented when the remote repository guarantees that for a fixed request to the remote, the same dataset will be returned. Management of the cache, such as file expiry or disk usage limits, is an implementation detail. It is expected that this be implemented using shared infrastructure by the system doing the retrieval, rather than being implemented as a local subset of a remote Data Repository. (UseCases: SQR10)

1.4.9 Local proxy

ID: DMS-MWBT-REQ-0055 (Priority: 1b)

Specification: It shall be possible to configure the Data Input system to use a local proxy to share remote retrievals of common Datasets.

Discussion: Thus allowing multiple local input requests (potentially from different users and systems) to use a shared cache. Note that this does not apply to the Data Output System (and may even be applicable only to read-only Data Input Systems) (UseCases: DRP17, DAX2)

1.4.10 Metadata association

ID: DMS-MWBT-REQ-0045 (Priority: 1a)

Specification: The Data Input System shall be able to associate observation/engineering metadata with a given Dataset.

Discussion: We will need to look at lightcurves and correlate with observation characteristics. This association will always be based on the date the given Dataset was observed. (UseCases:

COMM5)

1.4.11 Metadata merging

ID: DMS-MWBT-REQ-0035 (Priority: 1b)

Specification: It shall be possible to create an InMemoryDataset of a Dataset by gathering information from multiple distinct sources including a combination of files and databases.

Discussion: Create a Python object from multiple sources. This could be reading a FITS file from disk and augmenting the header information from a database query or from a separate header file. (UseCases: ARCH3)

1.4.12 Parameterized Subset of a Dataset

ID: DMS-MWBT-REQ-0033 (Priority: 1a)

Specification: It shall be possible to load into memory a parameterized subset of a Dataset without loading the full Dataset.

Discussion: An example of this could be reading a small postage stamp from a large image in a notebook where the notebook container has limited resource allocation. (UseCases: DRP18, DAX3, SQR15, COMM3)

1.4.13 Queries as Datasets

ID: DMS-MWBT-REQ-0057 (Priority: 1a)

Specification: The Data Input System shall support database queries as first class Datasets.

Discussion: This implies that the same Data Units and DatasetType may not return the same Dataset for all time. (UseCases: SQR5)

1.4.14 Reading persisted data

ID: DMS-MWBT-REQ-0032 (Priority: 1a)

Specification: The Data Input System shall be able to read any DataSet that has been written by the Data Output System using a Scientific Data Format.

Discussion: Here Scientific Data Format means a format that can be used as an intermediate file in processing such as FITS or HDF5. JPEG images are not included. (UseCases: ARCH2)

1.4.15 Remote Input DataRepository

ID: DMS-MWBT-REQ-0040 (Priority: 1a)

Specification: The Data Input System shall be able to read from non-local, non-POSIX input DataRepositories; this should include both database systems and file/object stores.

Discussion: This is not meant to preclude use of a local cache. E.g. if the backend is S3, the files need to be streamed to disk before they can be un-persisted. For example, this could be a Data I/O system that understands TAP and SIA VO protocols; or one that understands VOSpace or HDF5-in-the-cloud; or possibly S3-like object stores. (UseCases: SQR4, SQR12, SQR14, CI3)

1.4.16 Third party datasets

ID: DMS-MWBT-REQ-0048 (Priority: 1a)

Specification: It shall be possible for the Data Input System to read from catalogs provided by outside sources in the same API used for reading first class LSST datasets via a different plugin.

Discussion: We will need reference catalogs of all types (not just photometric and astrometric calibration). (UseCases: COMM9)

1.4.17 Consistent Interfaces for Input

1.4.17.1 Accessing official Data Releases

ID: DMS-MWBT-REQ-0037 (Priority: 1a)

Specification: The Data Input System interface shall provide access to official Data Releases from the LSST Science Platform.

Discussion: (UseCases: DRP2, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.4.17.2 Consistent Input Interface

ID: DMS-MWBT-REQ-0036 (Priority: 1a)

Specification: The Data Input System shall provide a consistent interface for loading Datasets into memory given a DatasetRef across different types of Data Repositories

Discussion: This is an interface expected by SuperTask execution, and we need to make it consistent in all contexts in which SuperTasks will be executed. The same interface may be used in (possibly interactive) analysis and validation work. (UseCases: DRP2, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.4.17.3 Outputs from notebook batch jobs

ID: DMS-MWBT-REQ-0038 (Priority: 1a)

Specification: The Data Input System interface shall provide access to the shared VOSpace file system that will contain the outputs of batch jobs launched from the Science Platform.

Discussion: The Notebook batch system will write data to a local node file system and these will be harvested by a process when the job completes to copy the data to the User VOSpace. (UseCases: DRP2, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.4.17.4 Outputs from test processing runs

ID: DMS-MWBT-REQ-0039 (Priority: 1a)

Specification: The Data Input System interface shall provide access to processing runs initiated for test/development purposes, from the same compute environment in which the processing was run.

Discussion: (UseCase: DRP2, DRP8, SCIVAL1, SCIVAL2, SCIVAL3)

1.4.18 Querying the EFD

1.4.18.1 Querying the Engineering and Facility Database

ID: DMS-MWBT-REQ-0041 (Priority: 1a)

Specification: The Data Input System shall be able to query specific subsets of the Engineering and Facility Database based on metadata from a visit.

Discussion: Calibration observations sometimes require extensive EFD data in high resolution covering the time of the visit. (UseCases: ARCH3, COMM2)

1.4.18.2 Read from the base EFD

ID: DMS-MWBT-REQ-0043 (Priority: 1a)

Specification: The Data Input System shall be able to read from the base EFD.

Discussion: The commissioning cluster will not necessarily have access to the transformed EFD. (UseCases: COMM12, COMM1)

1.4.18.3 Read from transformed EFD

ID: DMS-MWBT-REQ-0042 (Priority: 1a)

Specification: The Data Input System shall be able to read from the transformed engineering facilities database.

Discussion: (UseCases: COMM2)

1.4.18.4 Unified interface to summit/base EFD and transformed EFD

ID: DMS-MWBT-REQ-0044 (Priority: 1a)

Specification: Regardless of whether the Data Input System is reading from the raw or the transformed Engineering and Facilities Database, the interface (including arguments) from the pipelines perspective shall be the same.

Discussion: The same SuperTask needs to run at both the commissioning cluster using the raw EFD and on the commissioning archive with the transformed EFD. (UseCases: COMM2, COMM12)

1.4.19 Raw Data

1.4.19.1 Override part of a composite dataset

ID: DMS-MWBT-REQ-0051 (Priority: 1b)

Specification: It shall be possible to override part of a composite dataset with component datasets stored separately.

Discussion: This might be reading an updated WCS from the L1 prompt processing system on the night, or from a specific Data Release. Provenance for the WCS solution itself does not need to be included other than a unique identifier to allow provenance lookup. It could also be a photometric solution from a data release. (UseCases: ARCH3)

1.4.19.2 Reading raw data

ID: DMS-MWBT-REQ-0049 (Priority: 1a)

Specification: The Data Input System shall be able to read raw observation data files from the Archive Facility from the telescope and auxiliary telescope.

Discussion: These will be written in FITS format. (UseCases: ARCH3)

1.4.19.3 Reading up-to-date visit metadata

ID: DMS-MWBT-REQ-0050 (Priority: 1a)

Specification: The Data Input System shall be able to create an in-memory object from raw data, ensuring that this object contains up-to-date visit metadata.

Discussion: This could be new headers from the EFD that were not considered important when the observation was taken; or fixes to headers that were known to be incorrect after investigation (maybe a sensor was miscalibrated). Bulk download is not included in this requirement. (UseCases: ARCH3)

1.5 Data Output

1.5.1 Append to a DataRepository

ID: DMS-MWBT-REQ-0064 (Priority: 1a)

Specification: It shall be possible to add Datasets to a pre-existing Data Repository via additional processing.

Discussion: This is the situation where a user runs a SuperTask and then runs further processing on the output of the first. It is desirable for the output of the second processing task to be stored in the same Data Repository as the output of the first. (UseCases: COMM1, LDF1)

1.5.2 Blocked write operation

ID: DMS-MWBT-REQ-0073 (Priority: 1a)

Specification: A put operation on the Data Output System shall block until it has either worked or failed

Discussion: (UseCases: DRP22)

1.5.3 Combining composite datasets for export

ID: DMS-MWBT-REQ-0077 (Priority: 1b)

Specification: A facility shall be available to combine file-based composite datasets into a single file in a Scientific Data Format.

Discussion: For example, when downloading a PVI, the WCS solution, PSF and, possibly, provenance, components would be combined into a single HDF5 file for export to an external user. (UseCases: DAX9, ARCH5)

1.5.4 Consistent Output Interface

ID: DMS-MWBT-REQ-0067 (Priority: 1a)

Specification: The Data Output System shall provide a consistent interface for writing ConcreteDatasets to storage given a DatasetRef across different types of Data Repositories.

Discussion: This is an interface expected by SuperTask execution, and we need to make it consistent in all contexts in which SuperTasks will be executed. The same interface may be used in (possibly interactive) analysis and validation work. (UseCases: DRP3, SCIVAL1, SCIVAL2, SCIVAL3)

1.5.5 Creation of new DatasetTypes

ID: DMS-MWBT-REQ-0059 (Priority: 1a)

Specification: The Data Output system shall allow 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

Discussion: This allows persisting config and metadata from a new supertask or command-line task without editing any obs packages. It could also simplify repository configuration as many predefined dataset types could be specified as a much smaller of dataset prototypes. (UseCases: DRP12, SCIVAL1, AP3, SQR4, SQR6)

1.5.6 Data Output references

ID: DMS-MWBT-REQ-0075 (Priority: 1a)

Specification: The Data Output System shall give the Data Discovery System a full DatasetRef that can be used to later discover the DataSet that was just written.

Discussion: (UseCases: DRP3, SCIVAL1, SCIVAL2)

1.5.7 Filename invariance

ID: DMS-MWBT-REQ-0078 (Priority: 1b)

Specification: 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 shall also be unique and have a predictable name that is not dependent on data access mechanism.

Discussion: If a coadd FITS file is downloaded using the portal, it shall have the same name as if it was downloaded using VO access protocols. The file is allowed to have a different suffix if format translation has occurred. Additionally, if a composite is being merged into a single file (DMS-MWBT-REQ-0077), that file may have a different name to those stored internally.

It is not required to include directory hierarchy in this output name if that hierarchy is also encoded in the original filename. (UseCases: DAX9, ARCH5)

1.5.8 No clobber

ID: DMS-MWBT-REQ-0074 (Priority: 1a)

Specification: It shall be possible to configure the Data Output System such that it is an error if an attempt is made to persist a dataset that is already present in the output repository

Discussion: (UseCases: LDF1)

1.5.9 One Dataset to multiple output repositories

ID: DMS-MWBT-REQ-0063 (Priority: 1a)

Specification: It shall be possible for a single request to write a particular Dataset in more than one output repository, with the format used being different in each repository.

Discussion: This would allow an output FITS file to be written to one location and an HDF5 variant to be written to another. (UseCases: ARCH2)

1.5.10 Output location

ID: DMS-MWBT-REQ-0066 (Priority: 1a)

Specification: It shall be possible to configure the Data Ouput System via configuration to define output locations for outputs to POSIX file systems

Discussion: This will result in completely predictable output file paths. (UseCases: LDF1, SQR12, SQR1.5)

1.5.11 Output Staging

ID: DMS-MWBT-REQ-0079 (Priority: 2)

Specification: The Data Output System shall be able to transfer output Datasets produced by batch processing from temporary storage on compute nodes to persistent storage.

Discussion: (UseCases: DRP5, DRP6, SCIVAL1, SCIVAL2)

1.5.12 Outputs from Alert Production

ID: DMS-MWBT-REQ-0069 (Priority: 1a)

Specification: The Data Output System interface shall be usable by algorithmic code being run as part of a Alert Production.

Discussion: If algorithmic code always writes to a temporary location rather than a persistent archive, only writing to the temporary location needs to support the consistent interface. (UseCases: DRP3, SCIVAL1, SCIVAL2, SCIVAL3)

1.5.13 Outputs from Data Release Production

ID: DMS-MWBT-REQ-0068 (Priority: 1a)

Specification: The Data Output System interface shall be usable by algorithmic code being run as part of a Data Release Production.

Discussion: If algorithmic code always writes to a temporary location rather than a persistent archive, only writing to the temporary location needs to support the consistent interface. (UseCases: DRP3, SCIVAL1, SCIVAL2, SCIVAL3)

1.5.14 Outputs from Science Platform

ID: DMS-MWBT-REQ-0070 (Priority: 1a)

Specification: The Data Output System interface shall be usable by algorithmic code run in the Science Platform.

Discussion: This applies to both code run by a batch serves in the Science Platform and code run directly in a user process in the Science Platform. (UseCases: DRP3, SCIVAL1, SCIVAL2, SCIVAL3)

1.5.15 Outputs from test processing runs

ID: DMS-MWBT-REQ-0071 (Priority: 1a)

Specification: The Data Output System interface shall be usable by algorithmic code being run for test/development purposes, on both development compute environments at the archive center and in personal environments.

Discussion: (UseCases: DRP3, SCIVAL1, SCIVAL2, SCIVAL3)

1.5.16 Publishing to external microservices

ID: DMS-MWBT-REQ-0072 (Priority: 1a)

Specification: The Data Output System shall be able to publish to non-node-local micro services, via common web APIs.

Discussion: Measurements of metrics will need to be exported from the CI system and a good approach seems to be to publish the measurements using the QA microservice endpoints, via e.g. REST. (UseCases: CI2)

1.5.17 Remote Output DataRepositories

ID: DMS-MWBT-REQ-0065 (Priority: 1a)

Specification: The Data Ouput System shall be able to write to non-local, non-POSIX output DataRepositories; this should include both database systems and file/object stores.

Discussion: For example, this could be a Data I/O system that understands TAP and SIA VO protocols; or one that understands VOSpace or HDF5-in-the-cloud; or possibly S3-like object stores. (UseCases: SQR4, SQR12, SQR14, CI3)

1.5.18 Strong exception guarantee

ID: DMS-MWBT-REQ-0076 (Priority: 1a)

Specification: A put operation on the Data Output System shall provide the strong exception guarantee. If a put operation fails the previous state shall be restored.

Discussion: A put operation either works in full, or have no effect. In particular, if a dataset is a composite, all the parts must succeed, including any database writes. In particular, there is no guarantee that multiple puts are transactional. Purely private state may be modified by a failed put. (UseCases: DRP22)

1.5.19 Writer configurability

ID: DMS-MWBT-REQ-0060 (Priority: 1a)

Specification: The Data Output System shall be able to support local configuration of individual writer behavior.

Discussion: For example, enabling a FITS writer to use a specific data compression scheme. (UseCases: DRP26)

1.5.20 FITS Format

1.5.20.1 Writing FITS images

ID: DMS-MWBT-REQ-0061 (Priority: 1a)

Specification: The Data Output System shall be able to write in-memory image objects as FITS files.

Discussion: (UseCases: ARCH1)

Derived from Requirements:

DMS-REQ-0065: Provide Image Access Services

1.5.20.2 Writing FITS tables

ID: DMS-MWBT-REQ-0062 (Priority: 1a)

Specification: The Data Output System shall be able to write in-memory table objects as FITS files.

Discussion: (UseCases: ARCH1)

Derived from Requirements:

DMS-REQ-0078: Catalog Export Formats

1.6 Provenance Tracking

1.6.1 Dataset lookup: provenance driven

ID: DMS-MWBT-REQ-0095 (Priority: 1a)

Specification: The Data Output System and the Data Discovery System shall provide interfaces for recording and subsequently reporting (respectively) the Datasets that were used as inputs when creating a given Dataset.

Discussion: This may record either the Datasets that were predicted to be used as inputs, the Datasets that were actually used as inputs (a strict subset of those predicted to be used as inputs), or both; some systems may require both. For example, if a task is given 10 input datasets but only uses 9 then this information must be tracked. ”Trickle up provenance” for ”was derived from”. (UseCases: SQR1, SQR9)

1.6.2 Provenance in Datasets

ID: DMS-MWBT-REQ-0096 (Priority: 2)

Specification: The Data Output System shall persist provenance metadata relating to the immediate parents of the Dataset, when persisting to a Scientific Data Format.

Discussion: This could be implemented as a composite dataset so long as we have the ability to persist a composite dataset into a single entity. (UseCases: ARCH5)

1.6.3 Provenance to raw data

ID: DMS-MWBT-REQ-0093 (Priority: 2)

Specification: The Data Output System shall persist provenance information describing all the raw data IDs that contributed to this Dataset, when persisting to a Scientific Data Format.

Discussion: This enables the raw data to be determined from any file without an external lookup to a database server. This presumes that each observation/visit has a unique identifier. (UseCases: ARCH5)

1.6.4 Provenance tracing

ID: DMS-MWBT-REQ-0094 (Priority: 1a)

Specification: The Data Backbone shall contain provenance data that allows queries to report on all the Datasets that were created using a specific Dataset (where Datasets can be some combination of metadata and filenames).

Discussion: If a raw observation is later determined to be bad, all coadds created from that raw observation should be locatable and reprocessed. (UseCases: LDF1, LDF101)

2 Task Framework

2.1 SuperTask

The following requirements have been developed by the SuperTask working group as design requirements that derive from the requirements identified in the working group. We have concluded that writing down every requirement in a completely design-independent way is some cases a difficult and artificial exercise with limited benefits. With that in mind, we have still tried to avoid too much design-dependence. The most important place where design-dependence emerges is in the assumption of a Python API and not a purely command-line-oriented system.

The key motivating considerations were:

- Pythonic API
- Support of development and production through a common interface
- Ability to predict all inputs and outputs to support data staging and creation of "walled gardens" for production jobs
- Butler I/O
- A flexible system for representing the data-grouping constraints of each processing step as code in the same module as the implementation of the step (i.e., the grouping constraints of co-addition should be represented as part of a co-addition SuperTask)

2.1.1 Design Overview

2.1.1.1 Complete algorithmic work specification

ID: DMS-MWST-REQ-0001

Specification: The design shall provide 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.

Discussion: A Pipeline specification fully represents the transformations to be performed, but does not represent the specific data to which the transformation is to be applied.

2.1.1.2 Pipeline execution context

ID: DMS-MWST-REQ-0002

Specification: The design shall allow a given Pipeline specification to be used in both development and production contexts.

2.1.2 Supervisory Framework

2.1.2.1 Asynchronous data retrieval

ID: DMS-MWST-REQ-0030

Specification: It shall be possible for the two SuperTasks to run in parallel and cooperate with each other, such that task B can block waiting for data that A needs while A is busy doing something else, then A can use the data obtained by B.

Discussion: This will be used to provide DIA Objects to the alert generation Supertask (Use-Case: AP1.e)

2.1.2.2 Butler instantiation

ID: DMS-MWST-REQ-0023

Specification: The supervisory framework shall create and supply the Butler required to support the I/O that will be performed in the "Run" phase, for each unit of work.

2.1.2.3 Campaign specifications

ID: DMS-MWST-REQ-0027

Specification: The supervisory framework shall accept Pipeline "campaign" specifications including:

- Specifications of outputs to be produced from a universe of available inputs, with the Pipeline processing the minimal set of inputs required to make the outputs.
- (Possibly) Specifications of inputs to be processed, with the Pipeline producing all possible outputs deriving from these inputs.
- Specifications of both inputs and outputs, with the input specifications treated as restrictions on the universe of available inputs; i.e., "intersection" logic is applied.

Discussion: There are open questions about exactly what form the DataId specifications will take, what logic should be applied, and how it will be implemented, which the SuperTask WG believes can be best understood by working on a prototype implementation.

2.1.2.4 Execution logging mechanism

ID: DMS-MWST-REQ-0025

Specification: The supervisory framework shall set up the standard LSST logging mechanism for both the "Pre-flight" and "Run" phases.

Discussion: It is anticipated that different specializations of the framework may connect the logging output to different destinations.

2.1.2.5 Generating a DAG

ID: DMS-MWST-REQ-0021

Specification: The supervisory framework shall support the “Pre-flight” phase of execution of a Pipeline on a specified set of inputs and/or desired outputs, resulting in a DAG for the processing, with the nodes in the DAG being the units of work to be executed, each one being: the combination of one of the processing steps in the Pipeline with a complete list of the inputs and outputs for an invocation of that step in the ”Run” phase (specified as pairs of fully specified DataIds and Butler dataset types).

Discussion: A specific supervisory framework specalization is free to consolidate these units of work “vertically” (along the processing flow) and/or “horizontally”, for efficiency, as long as this is consistent with the DAG.

2.1.2.6 Mandatory supported specializations

ID: DMS-MWST-REQ-0019

Specification: The supervisory framework shall support specializations suitable for at least the following execution environments:

- Level 2 (DRP), CPP, and other non-real-time production.
- Level 1 near-real-time production.
- Interactive, command-line execution.
- Execution in a persistent server (e.g., to support the SUIT Portal).
- Automated CI and verification testing.
- (desirable) Execution from a Python prompt (e.g., in a notebook)

Discussion: The requirement for command-line execution provides a successor capability to CmdLineTask.

Execution within a Python environment is not a core requirement because it is always possible to create a subprocess to invoke the command-line specialization of the supervisory framework.

2.1.2.7 Multiple specializations of execution environments

ID: DMS-MWST-REQ-0018

Specification: The supervisory framework shall be designed to support the creation of multiple specializations for different execution environments.

Discussion: The ”supervisory framework” is an evolution of the ”Activator” concept from the original SuperTask design. Very likely we'll still use the word ”Activator” in the code - I still think it's evocative. However, we avoided it in the requirements to ward off the implication that the concept was completely unchanged - the separation of ”Pre-flight” from ”Run” phases is more explicit now following the WG's efforts.

2.1.2.8 Provenance configuration

ID: DMS-MWST-REQ-0026

Specification: (Assuming the originally proposed DM provenance mechanism, which was envisioned to collect more fine-grained information than is likely to be available from non-intrusive Butler instrumentation, is still part of the production baseline:) The supervisory framework shall perform whatever setup is required for the fine-grained provenance mechanism.

Discussion: Resolution of Butler dataset / SuperTask - level non-intrusive provenance versus – or in addition to – the originally proposed provenance mechanism was beyond the scope of the SuperTask WG. This question should be addressed soon by DM.

2.1.2.9 Provenance discovery

ID: DMS-MWST-REQ-0024

Specification: (desirable) The supervisory framework shall provide for “non-intrusive provenance” discovery, tracking the actual execution (in the ”Run” phase) of units of processing on input datasets, their outputs, and their associated Task structure and configuration.

Discussion: This is intended to be done via instrumentation of Butler calls. The recording mechanism is TBD. The information so collected is complementary to the predictions of inputs and outputs from the ”Pre-flight” phase. The production system is not expected to preserve this information.

2.1.2.10 Serialization of workflow DAG

ID: DMS-MWST-REQ-0022

Specification: The supervisory framework shall provide a serialization form for the results of pre-flight, so that they can be computed in one process and executed under the control of one or more others.

Discussion: Community tools for expressing workflows, such as the Common Workflow Language, will be evaluated for use in this serialization.

2.1.2.11 Standardized framework implementation

ID: DMS-MWST-REQ-0020

Specification: The supervisory framework shall provide a common implementation of the logic required for interpretation of the Pipeline steps and their data groupings (and thus the possible parallelizations).

Discussion: The intent is that this logic would be applied in all specializations, so that the execution pattern (though not necessary the actual parallelizations) would be consistent.

2.1.3 Pipeline Specification

2.1.3.1 Butler dataset type configuration

ID: DMS-MWST-REQ-0014

Specification: The Pipeline APIs shall provide for the use of the configuration mechanism to control the Butler dataset types used for input and output by each processing step.

Discussion: The use of string constants in Butler.get() calls will be replaced by the use of values of dataset-type configuration fields. (This is one of the more design-specific requirements in the list.)

2.1.3.2 Butler instances

ID: DMS-MWST-REQ-0009

Specification: The API for the execution of an individual processing step on specific data shall allow the caller to supply a Butler instance for the step's use.

Discussion: Steps (i.e., SuperTasks) are expected to generally perform I/O via a Butler, using Butler.get() to obtain inputs for Tasks' run() methods and Butler.put() to output the results produced by Tasks.

As in current Science Pipelines usage, Tasks are assumed to operate solely on Python-domain objects supplied as arguments to their run() methods, with results returned through a pipe.base.Struct return value. This is sometimes called the ”no I/O in Tasks rule”.

Therefore, in this design, the Butler calls are expected to occur at the SuperTask level, above the Task level of the call tree.

2.1.3.3 Changes of parallelization

ID: DMS-MWST-REQ-0007

Specification: A Pipeline specification shall permit each step in a sequence to have a different required data grouping, and therefore an implied change of permissible parallelization from each step to the next.

Discussion: For example, a Pipeline could include a 1:1 step performing single-exposure calibration and characterization, an N:1 step performing coaddition by tract, patch, and filter, and a 1:1 step producing a source catalog for each coadded tile.

2.1.3.4 Dataset grouping

ID: DMS-MWST-REQ-0006

Specification: A Pipeline specification shall specify how datasets must be grouped for each step in the sequence.

Discussion: For example, this would cover the grouping of inputs to a coaddition SuperTask by filter band and sky tile (tract and patch).

2.1.3.5 I/O via Butler

ID: DMS-MWST-REQ-0013

Specification: The Pipeline specification APIs used in the ”Run” phase shall provide for a Butler instance (provided by the supervisory framework) to perform all required I/O for each step in the “run” phase.

Discussion: Some exceptional cases requiring direct I/O to databases may be excluded from this restriction. (E.g., to permit database ingest itself to be handled in this framework.)

2.1.3.6 Implied inputs

ID: DMS-MWST-REQ-0012

Specification: The design shall include APIs that support resolution of full DataId specifications for “implied inputs” such as calibration frames, reference catalog shards, etc. This resolution shall be possible at the ”Pre-flight” stage, so that the true identities of ”implied inputs” are known and exhibited in the DAG.

2.1.3.7 Phases of execution

ID: DMS-MWST-REQ-0011

Specification: The design shall include Pipeline and step APIs that support “Pre-flight” and “Run” phases of execution organized by the supervisory framework. These are further constrained in the next section; the basic definition is:

- Pre-flight: support the computation of a DAG for the application of a Pipeline to a specification of input and/or output datasets.
- Run: invoke the units of work defined in the DAG (a unit of work is a pair of a processing step with its input and/or output DataIds).

Discussion: The DAG produced by the ”Pre-flight” phase is then able to be analyzed by the supervisory framework to determine how to batch up and/or parallelize units of work for actual execution in the ”Run” phase.

The information obtained at the ”Pre-flight” phase also allows the offline production workflow system to set up ”walled gardens” for individual SuperTask executions in the ”Run” phase, to which only the identified inputs are staged.

The ”Pre-flight” phase is permitted to predict the use of a superset of the inputs that end up actually being required in the ”Run” phase, e.g., if estimates of the sky-tile coverage going into a coaddition stage are not quite accurate because the true WCS is not known until processing occurs. It is understood that this is intended to be a best effort to get close to the true requirements - as a matter of quality-of-implementation SuperTasks should not carelessly inflate their predictions.

2.1.3.8 Pipeline configuration

ID: DMS-MWST-REQ-0005

Specification: A Pipeline specification shall specify the configurations of all the units of code to be run, using the existing LSST stack “pex_config” mechanism.

Discussion: SuperTasks will have pex_config configurations of their own in addition to the configurations of the Tasks they contain (see below).

Derived from Requirements:

DMS-REQ-0306: Task Configuration

2.1.3.9 Pipeline specification

ID: DMS-MWST-REQ-0004

Specification: A Pipeline specification shall specify the units of code to be run and a sequence in which they are to be run.

Discussion: The sequence specification need only be a explicit ordered list. It is not required to support looping, branching, or step-skipping.

The ”units of code” are the SuperTasks.

2.1.3.10 Pipeline specification definition

ID: DMS-MWST-REQ-0017

Specification: (Precise nature of the mechanism for constructing a Pipeline specification)

Discussion: An open design question is whether Pipelines will be constructed purely in Python, e.g., by writing code that instantiates the necessary SuperTasks and inserts them into a Pipeline object, or from a specification in a configuration language (e.g., YAML) that is then interpreted by a centrally-provided function and converted into the appropriate composite of Python objects.

2.1.3.11 Pre-execution overrides

ID: DMS-MWST-REQ-0016

Specification: The Pipeline (and supervisory framework) design shall support pre-execution (before the ”Pre-flight” phase) programmatic overrides to the configurations specified for a Pipeline. Such overrides must be capable of being captured for purposes of provenance recording.

Discussion: These overrides are a generalization of the command-line overrides provided in the existing CmdLineTask mechanism. It must continue to be possible to capture the full run-time configuration as a snapshot. See also related requirement DMS-MWBT-REQ-0087.

2.1.3.12 Programmatic insertions

ID: DMS-MWST-REQ-0015

Specification: The Pipeline design shall support programmatic insertions (before the ”Pre-flight” phase) of additional processing steps to an already-specified Pipeline’s processing sequence. The provenance mechanism (see below) must be capable of capturing these additional steps.

Discussion: The intent is that this interface could be used by a supervisory framework to add, e.g., quality analysis or other monitoring steps.

2.1.3.13 Programming API

ID: DMS-MWST-REQ-0003

Specification: The Pipeline specification interface shall be available as a Python API.

2.1.3.14 Supervisory framework

ID: DMS-MWST-REQ-0010

Specification: The design shall permit a supervisory framework (see below) to execute any Pipeline, based solely on information obtained programmatically from the Pipeline specification, against a data specification (see below).

2.1.3.15 Task memoization

ID: DMS-MWST-REQ-0031

Specification: A properly configured SuperTask executed with the same datasetRefs, configuration, and code shall return a ”cached” version of the result (a datasetRef to the dataset, or the dataset itself whichever is appropriate) instead of redoing the computation. It shall be possible to configure the SuperTask to turn memoization off. This would produce an error if an output dataset exists.

Discussion: SuperTasks should contain enough internal state to memoize the execution methods. This allows notebooks in the LSP to efficiently use limited computation resources. This is a SuperTask requirement. (UseCase: SQR10)

2.1.3.16 Use of Tasks and configurations

ID: DMS-MWST-REQ-0008

Specification: A Pipeline specification shall support the organization of work within a step in terms of Tasks, and shall supply the configurations the Tasks require.

Discussion: A SuperTask is assumed to be built from Tasks, and uses the existing hierarchical configuration mechanism for Tasks and sub-Tasks to represent their configuration.

2.1.4 Performance Requirements

2.1.4.1 Alert and DIA Object transmission rate

ID: DMS-MWST-REQ-0029

Specification: It shall be possible for a SuperTask to send at least (**nAlertVisitAvg**/number of science CCDs) alerts to the alert distribution system, plus the same number of DIA Objects and DIA Sources to the L1 database, all within **timeToIssueAlerts** seconds. The goal is to send this information via the Butler, but a more direct path is acceptable if needed.

Discussion: The time limit is based on a timing diagram by K-T Lim, which is the best guess we have in lieu of a formal timing breakdown. In that diagram the time allocated for generating alerts is approximately 10 seconds. I arbitrarily allocated half of that for computing the information (e.g. updating DIA Objects and creating alerts) and half for transmitting it. (UseCase: AP1.f)

Description	Value	Unit	Name
Minimum number of alerts required to be accommodated from a single standard visit	10000	integer	nAlertVisitAvg
Time allocated for issuing alerts and storing them for later retrieval.	5	second	timeToIssueAlerts

2.1.4.2 Round trip time for DIA Sources and DIA Objects

ID: DMS-MWST-REQ-0028

Specification: The total time for one Supertask to start writing DIA Objects and DIA Sources to the L1 database and another Supertask to query and finish retrieving those same products for the next field must be less than **timeToWriteReadAlertsDB** seconds. Supertask will use the

Butler for the put, query and get, unless alternatives are required for adequate performance. This time should be divided between the put and the query+get, to make two requirements, with some slop between them.

Discussion: The time from the start of one visit exposure to the start of the next is 35 seconds. The budgeted time for source association is roughly 9 seconds. That leaves roughly 26 seconds if we are to use DIA Objects from one visit for processing the next visit. (UseCase: AP1.e)

Description	Value	Unit	Name
Maximum time allowed between one SuperTask storing alerts in the database and another SuperTask reading those alerts from the database.	25	second	timeToWriteReadAlertsDB

3 Pending

3.1 LSST Data Facility

These requirements came out of the LDM-592 Use Case analysis but should be handled in an LSST Data Facility requirements document, where they would receive a proper ID.

3.1.1 Execution on archived data

ID: 16 (Priority: 2)

Specification: The LDF shall make it possible to request data that may have been archived and have it retrieved asynchronously and made available in a user's workspace.

Discussion: The remote storage could include tape storage. This differs from DMS-MWBT-REQ-0058 in that the Datasets are transferred explicitly in advance, rather than as-needed, and it differs from DMS-MWBT-REQ-0052 in that the targeted storage is a full data repository on at least semi-persistent storage, rather than a scratch area. (UseCases: SQR16)

3.1.2 Read from development shared persistent storage

ID: 15 (Priority: 1a)

Specification: The LDF shall make it possible for privileged users to be able to run Super-Tasks that read directly from curated datasets held in the development provenance enabled, shared, persistent storage area.

Discussion: The CI system won't be like the regular batch system. Instead it will be running a few jobs concurrently on well known datasets. (UseCases: CI1)

3.1.3 Write to development shared persistent storage

ID: 14 (Priority: 1a)

Specification: The LDF shall make it possible, given sufficient permissions, do output to a persistent, shared, provenance enabled storage space all information supplied while running a pipeline, including writes of provenance information, files, and database rows if required.

Discussion: The outputs of historical reductions will need to be curated and readily available. (UseCases: CI1)

References

[1] **[LDM-592]**, Jenness, T., Bosch, J., Gower, M., et al., 2017, *Data Access Use Cases*, LDM-592, URL `https://ls.st/LDM-592`