SUI / Level 3 and SuperTask feedback on the Butler

For reference... our view of the get()/Read capabilities currently provided or desired to be provided by the Butler:

1. Reconstitution of Python-domain objects
2. Abstraction of storage (file formats, storage mechanisms, and naming) - currently only really abstracts naming
3. Understanding of associations between datasets (e.g., temporal and spatial)
    - Find synthetic flat appropriate to a raw image
    - Find calibrated images contributing to a coadd patch
4. Wildcarding (partially specified DataIds)

Capability (1) must by definition be provided by a Python language API. The others could be provided as a service. See below.

SUI feedback:

1. When the Butler returns multiple results (e.g., from a wildcard or a 1:N association) we would like the results to be available as a list of references in a form directly (or at least requiring only a trivial transformation) usable with the LSST data retrieval Web APIs (DAX). The wildcard and 1:N resolution are useful in the SUI but we then need to be able to operate on the result (which may be lengthy) at the metadata table level.
    a. Unknown User (npease):
        i. DM-4548 - Jira project doesn't exist or you don't have permission to view it.
        ii. also DM-4549 - Jira project doesn't exist or you don't have permission to view it.
2. We would like API support for the ability to retrieve a dataset in multiple Python forms - e.g., we would like to be able to support retrieval of results in community-standard forms (e.g., pandas, astropy tables) as well as their "native" LSST forms – where these are different.
    a. Unknown User (npease): DM-4551 - Jira project doesn't exist or you don't have permission to view it.
3. We need to understand how put()/writing works when multiple repositories are made visible through a single Butler. For get()/reading a single search order makes sense. For put() it may be desirable to support alternative destinations (local disk, user workspace, Level 3 DB) or even multiple destinations for a single put().
    a. Unknown User (npease): proposed spec in CLO
4. We need to understand how authentication information is passed through the Butler to background services it may access.
    a. Unknown User (npease): DM-4552 - Jira project doesn't exist or you don't have permission to view it.
5. We would be very interested in a "remote Butler" functionality, in which capabilities 2, 3, and 4 above are provided by a well-defined Web API, allowing capability 1 to be provided by a thinner Python wrapper around a Web API call. This would be useful directly to non-Python components of the SUI, as well as making it easier to provide Butler functionality to Level 3 users running remotely. In principle, this could be done by ensuring that the capabilities exposed by the DAX interfaces include Butler capabilities 2, 3, and 4 – or the DAX interfaces could be a lower level, providing capability 2 and perhaps capability 4, for specific repositories, with the "remote Butler" providing capability 3.
    a. Unknown User (npease): DM-4554 - Jira project doesn't exist or you don't have permission to view it.
6. Do get and put operations need to take versioning of an entity into account?

SuperTask feedback:

1. We would like to decouple put() calls from persistence, allowing deferred writes and/or administrative control of whether put() just generates an in-memory temporary or actually writes to persistent store.
    a. Unknown User (npease): does allowing the in-memory destination (in

    DM-4542 - Jira project doesn't exist or you don't have permission to view it.

) satisfy this need? If not we should have a voice meeting about this to discuss details.
2. Thinking about the execution interface of the SuperTask, we believe we need something more sophisticated than just DataRef as a layer above the Butler.