Gen3 DataId and DatasetRef serialization

(Draft of an RFC.)

Proposal

Adopt a canonical serialization for DataId and DatasetRef objects.

Motivation and Use Cases

To facilitate reliable exchange of information "outside of code", primarily through the human world and in copy/paste operations, about specific datasets.

To avoid requiring consumers of results from dataset metadata queries to have to rebuild `DataIds` by assembling pieces from multiple columns returned from the query.

1. When performing an ObsTAP or SIAv2 query against LSST data, the availability of a string serialization of `DatasetRef` values for the datasets reported in the query result would enable a `DatasetRef`-valued column to be added to the query result, marked-up with data model metadata that enables a programmatic or UI client to recognize the column as such. This in turn would enable the following applications:
    a. For the case of programmatic queries, LSST can provide a Python query API that directly returns DatasetRef or DataId objects, as appropriate, which can then be used directly in Python code to access and process the datasets in question.
    b. For the case of UI-driven (Portal Aspect) queries, the Portal can provide UI actions such as "copy reference to clipboard" that facilitate interoperation between the Portal and Notebook Aspects - the user can search for data in the Portal and then paste immediately usable references to that data directly into Python code in the Notebook.
    c. A programmatic or UI client can use IVOA DataLink metadata in a query result to discover additional resources related to a row returned from a dataset metadata query. See (2) below for more detail.
    d. By being able to *recognize* a column as containing a DatasetRef serialization, a UI client could display it with space-efficient UI elements like a DATASETREF icon with a copy-to-clipboard function, instead of displaying a cryptic JSON string in a wide column.
2. One or more IVOA DataLink "{links} services" can be created whose `ID=` query parameter takes a DataId or DatasetRef serialization and returns references to related data. This could be used for things like linking a calexp to the directly associated raw, difference image, etc., and it could also be used to look up indirect associations like going from a visit image to the calibration images that are configured to be the appropriate ones for it.
3. In automatically-generated, or even human-generated, reports, datasets that had processing problems or were otherwise worth noting could be identified including the canonical serialization, facilitating readers going to a Python prompt to perform further analysis on that dataset.

Serialization technology

It is proposed to use JSON-LD (standard | Wikipedia) for the canonical serialization.

This means that the data content of a DataId or a DatasetRef will be represented as JSON, but in addition that the JSON will include type information referenced to a vocabulary published by LSST.

A typical JSON-LD object might look like this (hat tip to Brian Van Klaveren):

```
{
  "@context": {
    "@vocab": "http://lsst.org/butler-dm/v3/",
  },
  "@type": "DataId",
  [all other JSON here]
}
```

where `http://lsst.org/butler-dm/v3` defines the root of a vocabulary of relevant terms, such as DatasetRef, DataId, Dimension, etc.

In cases where many serialized values must be transmitted, e.g., as a column of `DatasetRef` values in a serialized table, we envision using table-level metadata to define the column type in such a way that consumers of the table can retrieve the JSON-LD type information, and limiting the JSON text of each row's value in the column to the actual data content. A client of the serialized table (e.g., a Python API wrapping an HTTP query, or a UI client) can use the values directly, or re-wrap the values with the type information if it is appropriate to regenerate the full JSON-LD object.

For a VOTable serialization, we will work with the IVOA to devise a specific proposal for the content of the `<FIELD>` element of the VOTable header that would realize this idea. We anticipate this may mean using either a special *utype* value or UCD to indicate JSON-LD-typed data.

The Felis specification language for the LSST data model is already based on JSON-LD and thus would naturally accommodate the inclusion of JSON-LD-typed columns in tables defined in that data model. We expect this capability to be used to define `DataId` and/or `DatasetRef` column types in the dataset metadata tables exposed via ObsTAP and/or SIAv2 services.

Open questions

Representation of both types of DataId

Will there be distinct types for the "minimal" and "fully expanded" `DataId` types introduced in

DM-17023 - Jira project doesn't exist or you don't have permission to view it.

?

Exact content of the serialization

What attributes of DataId and DatasetRef will be persisted? (E.g., will any of the producer / consumer / run information be persisted?)