These notes pertain to the interface between CRDS and the archive catalog.

Nominally the catalog team will produce web services to be used for fetching dataset parameters CRDS uses to associate reference files with datasets. CRDS uses these parameter sets to reevaluate reference assignments as new reference files arrive in order to determine which datasets should be reprocessed. A more advanced feature of CRDS is to simulate reference table row lookups for the same purpose, to determine how a dataset's specific table rows change and drive reprocessing needs.

I am trying to obtain web services or catalog access funtionally equivalent to these strawman RPC calls:

get_dataset_ids_by_instrument(instrument)
→ [dataset_id, …]

inputs:
instrument: instrument name string
e.g. “MIRI”

returns:
dataset_ids: list of all compound dataset ids for instrument

e.g. ['J59L54010:J59L54GEQ', ...] (associated)
['LA9K02DGQ:LA9K02DGQ', …] (unassociated)

Abstract id “<product_id>:<exposure_id>”,

dataset_ids can be mixture of associated and unassociated exposures.

get_dataset_headers_by_ids(dataset_ids, requested_parameters)

→ { dataset_id : { dataset_parameter : value, …}, …}

inputs:
requested_dataset_ids: list of dataset id strings

requested_parameters: list of parameter name strings

e.g. [“DETECTOR”, “FILTER1”, ..]

returns:
returns a list of all compound dataset ids for instrument

e.g. {'J59L54010:J59L54GEQ' : { “FILTER1” : “CLEAR1L”,
“FILTER2” : 'F814W',
…,
}

As I did this for CRDS, I limited the header RPC to 5000 ids at once and partition the ids.

There are currently two fluid standards for CRDS reference assignment and dataset reprocessing parameters. There are unfinished specs being developed by the pipeline working group (nominally Van Dixon), and there are the defacto CRDS rules which have been defined by JWST calibration developers.

Quoting an e-mail from Van Dixon prompted by Mike asking for an equvalent to HST's ICD-47:

> We are slowly writing such a document. Take a look at the Confluence page for the PWG. Under "Calibration Reference Files" you will see…
>
> **Calibration Reference Files**
>
> - JWST Calibration Reference Files: File Formats for the Build 3 Pipeline – Reference files for Build 3 should comply with the formats specified in this document.
> - JWST Calibration Reference Files: File Formats for the Build 4 Pipeline – Document under construction. Do not use!
>
> The first version of the document describes the reference files and formats expected by the Build 3 pipeline. It has the advantage of being static and the disadvantage of being obsolete. The second version of the document is under constant modification and will eventually describe the Build 4 files and formats. Todd should use the Build 3 version of the document, but he should be aware of the Build 4 version. In particular, the keyword errors that Rossy discovered back in November have been corrected in the Build 4 version, but not in the Build 3.

A subtle distinction between Van's comments and our task has to do with reference files versus dataset files. Nominally all these reprocessing parameters come from dataset files, not references. Generally there is a 1:1 correspondence, but there is no guarantee that all reference file parameters will be used, that the spec list is complete, or even that corresponding dataset parameters are named the same way between references and datasets. References and datasets are opposite sides of the reference file assignment coin, and this task is really about dataset parameters. I think the specs are about reference files, not datasets.

A thing to keep in mind with CRDS is that the required parameters will change over time, but that should be covered by CRDS requesting specific parameter names and back-filling newly added parameters in the catalog as required. Since CRDS supports many versions of rules, once a parameter is in the catalog, it should probably stay there permanently so CRDS can reevaluate historical sets of rules.

The CRDS web site for JWST is here: https://jwst-crds.stsci.edu/

The latest version of CRDS rules are shown as Operational References. Digging down to a particular reference type, the column names correspond to the lookup parameters used for reference file assignments. Table row lookup parameters are not shown but we need to plan for those too. The website drops the common "META." prefix for brevity but the web service should not.

Eample JWST parameter set as defined in CRDS:

```
{'fgs': ['META.INSTRUMENT.NAME',
  'META.INSTRUMENT.DETECTOR',
  'META.INSTRUMENT.FILTER',
  ],
 'miri': ['META.INSTRUMENT.NAME',
  'META.EXPOSURE.READPATT',
  'META.EXPOSURE.TYPE',
  'META.INSTRUMENT.BAND',
  'META.INSTRUMENT.CHANNEL',
  'META.INSTRUMENT.DETECTOR',
  'META.INSTRUMENT.FILTER',
  'META.SUBARRAY.NAME',
  ],
 'nircam': ['META.INSTRUMENT.NAME',
  'META.INSTRUMENT.DETECTOR',
  'META.INSTRUMENT.FILTER',
  'META.INSTRUMENT.PUPIL',
  'META.SUBARRAY.NAME',
  ],
 'niriss': ['META.INSTRUMENT.NAME',
  'META.INSTRUMENT.DETECTOR',
  'META.INSTRUMENT.FILTER',
  'META.SUBARRAY.NAME',
  ],
 'nirspec': ['META.INSTRUMENT.NAME',
  'META.EXPOSURE.TYPE',
  'META.INSTRUMENT.DETECTOR',
  'META.INSTRUMENT.FILTER',
  'META.INSTRUMENT.GRATING',
  'META.SUBARRAY.NAME',
  ]}
```

Practically speaking, this is an example only, it is unavoidably subject to the following areas of change:

1. Changes in parameters

2. Change in specification of parameter names

3. I wouldn't rule out changes in instrument naming and organization

4. Changes to follow the evolving specs and actual code development.

So, the above are the CRDS defacto parameters, not anything official beyond “this is what the code actually does.” There is still some divergence from the buld 3 spec. The build 4 spec specifically says “don't implement this yet” but we should consider that as a target.

Nitty gritty issues:

Sooner or later we'll hit these I think.

Service Protocol

One important aspect of any web service will be the protocol used for remote function calls. In CRDS, so far I've used Django JSONRPC which is quite easy. The Python / Apache / Django configuration used in CRDS is one approach to getting JSONRPC, but JSONRPC is language agnostic and has Java support as well. I mention this protocol as one which has worked for me and an option to consider.

TBD web service protocol, CRDS services are currently JSONRPC

TBD protocol exception / error handlng: JSONRPC response has an "error" status field

TBD protocol version, JSONRPC has several versions like 1.0, 1.1, 2.0, I'd guess 1.0 is fine.

JSONRPC references:

http://json-rpc.org/
http://json-rpc.org/wiki/specification
http://json-rpc.org/wiki/implementations

Parameter Keywords

Issues related to keywords are such as:

What are they?

Nominally keywords are FITS header parameter names which describe instrument configuration (e.g. INSTRUME, FILTER) or possible work-flow (e.g. yyyyyCORR) keywords used to assign reference files. Additional keywords may be those used by calibration software to look up specific rows within tables.

CRDS itself is currently using JWST data model names in the CRDS rules to refer to matching parameters. The JWST data model is part of the calibration software. As an example, in the data model "META.INSTRUMENT.NAME" maps onto the FITS key "INSTRUME". The CRDS rules ask for the parameter as "META.INSTRUMENT.NAME" from the data model, isolating CRDS from the reference file or dataset file format. There are definite plans in SSB to use reference file formats other than .fits, nominally the .asdf format.

Where do they come from?

The question here is how the pipeline obtains the parameters it catalogs. If the pipeline fetches them by reading dataset files, there may be an advantage to using the JWST data model in the pipeline. That would insulate the pipeline from most changes required to support new data formats which are in the works. So (I think) there may be a mapping here, which is basically from <dataset + parameter name> : <table + row + column>. The mapping is used to fetch the parameter value from someplace, and store it in a catalog. There's a question about how to express <parameter name>, nominally as

FITS keyword or JWST data model dotted path. If <parameter name> is used to access dataset files, then be aware that the underlying data file format may change, so now may be the time to adopt the data model in the pipeline rather than later.

Parameter Values

Not a biggie, but for simplicity and uniformity all parameter values currently used by CRDS are strings, even those which are apparently numbers. Another thing to consider for other services is structured parameters, JSONRPC and JSON supports nested objects so richer values are easily possible with that protocol.