Observational Metadata Data Structures and Extraction

We have a number of observational metadata data structures and extraction-from-FITS scattered around the stack, and I'd like to get them reorganized and talking to each other. Ideally this would be done in a way that permits at least some existing Gen2 code and all Gen3 code to use the new APIs, and be done fast enough for at least some Gen3 code to never have to deal with some aspects of the current state of affairs.

To avoid confusion below, when I say "Exposure", I mean the daf_butler/RFC-484 concept, not the afw.image class; I will always say "afw.image.Exposure" to mean the latter.

Relevant Jira Issues

⚠ DM-15362 - Jira project doesn't exist or you don't have permission to view it.

⚠ RFC-484 - Jira project doesn't exist or you don't have permission to view it.

⚠ DM-15646 - Jira project doesn't exist or you don't have permission to view it.

Data Structures: Current State

daf.base.PropertyList/PropertySet

Flexible; what we use to read in FITS headers before doing any extraction, and what we attach to afw.image.Exposure to hold anything we didn't extract.

If I had to do it all over again, I'd want to try to use astropy.io.fits.Header objects instead. We can't easily switch to that now because we need this data structure in C++, but it'd be nice to keep that door open where possible.

afw.image.VisitInfo

Our primary object for structured, plain-old-data (POD) observational metadata. Also attached to afw.image.Exposure. At least most of this information is about an Exposure, not a Visit, so in that light perhaps it should be renamed (but ExposureInfo is taken, because of afw.image.Exposure; grr). Immutable, which is what we want in a lot of contexts, but a big inconvenience in the low-level code that needs to create it.

afw.image.ExposureInfo component classes

There are several afw non-POD objects that often have FITS-metadata persistence: SkyWcs, Filter, [Photo]Calib, Box2I.

Gen2 Registry Tables

These are totally camera-dependent bits of observational metadata, usually denormalized to have Exposure/Visit information duplicated for each Sensor. Includes Gen2 Data IDs.

Gen3 Exposure and Visit DataUnit Entries

These are represented in-memory as Python dicts and in SQL as rows in two different tables. The schemas of these were inexpertly derived from what VisitInfo has, and the breakdown between what's in Exposure and what's in Visit is only slightly better than arbitrary. But I think the *idea* of separating them is sound, even if that results in some duplication.

Metadata Extraction: Current State

obs.base.MakeRawVisitInfo

A (conceptual) abstract base class that other obs packages override to support attaching a VisitInfo instance to a raw afw.image.Exposure. As of DM-15189, also used to construct Exposure and Visit DataUnit entries in Gen3 ingest. Always strips entries from the input PropertyList. The closest thing we have to a general metadata-extraction API, but customization points are quite opaque and it only covers what's in VisitInfo.

afw.image.ExposureInfo component classes

These typically (but not always) have functions to construct themselves from a PropertyList and populate a PropertyList with their FITS-metadata representation. But those methods have inconsistent names, signatures, and behavior w.r.t. whether or not they strip extracted keys from the PropertyList.

Gen2 Raw Ingest

pipe.tasks.IngestTask (and its subtasks and per-obs derived classes). Populates Gen2 Registry tables, including extracting Data IDs from headers. A very flexible system, but because the outputs are not standardized across cameras, it's not the kind of system we want in Gen3 or any other context where the output is a predefined set of fields.

Gen3 Raw Ingest

obs.base.gen3.RawIngestTask and its subtasks. Extracts information to populate Gen3 DataUnit entries by delegating to MakeRawVisitInfo. Cannot delegate extracting Data IDs to MakeRawVisitInfo, as it doens't do IDs - instead I've copied and slightly modified logic from concrete Gen2 IngestTasks (just HSC for now).

Data Structures: Design Goals

Successor-to-VisitInfo

A pair of classes to replace VisitInfo whose fields map exactly to the Gen3 schema for Exposure and Visit. This does not imply VisitInfo is what should change more in terms of the actual fields - it's probably better as-is than the Gen3 schema - what matters is that they are made consistent.

In Python, fields of these objects should be properties/attributes, not getters. It should be possible to freeze these objects so they cannot be further modified (which may be implemented by making the classes immutable - see below).

I currently think it's probably good for these to also include the Data ID key-value pairs that are present in the Gen3 Exposure and Visit concepts, but I could be persuaded otherwise (and it's quite likely we'll want to extract those into a sub-object).

Prototyping these in Python would make a lot of sense, but I think we'll ultimately need a them to be wrapped C++ so we can attach them to afw.image.Exposure instances.

afw.image.ExposureInfo component classes

Changing these classes is out of scope for this project, aside from their extraction methods (see below).

Gen2 Registries

We should not try to change Gen2 registry schemas; we'll just wait a bit longer for them to be replaced wholesale.

Gen3 Exposure and Visit DataUnit Entries

These need to be kept in sync with Successor-to-VisitInfo.

Metadata Extraction: Design Goals

Successor-to-MakeRawVisitInfo

A unified mechanism for constructing the Successor-to-VisitInfo from raw data across different Cameras.

This should be be a true ABC with clear customization points for each field in Successor-to-VisitInfo, with default implementations that delegate to the extractors for other fields as appropriate.

It must be possible to configure this to leave the input PropertyList unchanged, to strip all consumed entries, or to consume entries without actually extracting metadata (or at least not wasting time doing expensive things when all the caller wants is to strip would-be-used entries).   Note that this is tricky, because many header entries (e.g. NAXISn) are probably used by many output fields, and can only be stripped after all of them are extracted.

It would probably be most convenient to have this incrementally build a successor-to-VisitInfo (which has implications for whether it can be intrinsically immutable), but I think the important constraint is that the thing incrementally built must duck very much like one.

It should also be possible to ask for only some metadata to be extracted and stripped (e.g. only the Data ID, or only Visit metadata, not Exposure metadata). As with strip-only mode, it of course doesn't matter if if it actually extracts more than this, as long as it doesn't take too long *and* it only strips the entries corresponding to what was requested.

For bonus points, the base class should hide whether the data structure being operated on is a PropertyList or something else (e.g. astropy.io.fits.Header), so we could change this in the future without having to rewrite per-obs customizations. Not worth it if this leads to writing another abstraction layer over a flexible metadata dictionaries.

Can be implemented entirely in Python.

afw.image.ExposureInfo component classes

The methods to build from metadata and extract from metadata that currently exist should be standardized in name, signature, and behavior, as much as possible. In some cases, it may not be appropriate to make things identical - `bboxFromMetadata`, for example, probably has to remain a free function rather than a Box method, because Box may not want to depend on PropertyList. But the differences should be minimized, and those that should remain should be due to actual constraints, not accidents of history.

While it should not necessarily be the responsibility of Successor-to-MakeRawVisitInfo to construct all of these objects, it should be possible to have the same control over stripping (including stripping shared entries) when jointly using any combination of component-class extractors and Successor-to-MakeRawVisitInfo.

Gen2 Raw Ingest

We shouldn't try to make many changes here - at most, we'd want to start delegating to Successor-to-MakeRawVisitInfo to extract Data IDs.

Gen3 Raw Ingest

This should be rewritten to use Successor-to-MakeRawVisitInfo for essentially all metadata extraction.