PAL Versioning Requirements Draft 2

Definition of terms

A new ***version*** of an entity is an instance of an entity which is considered to be an *improvement* of a previous version of an entity.

> For example, SPIRE have an entity (in this case, a product) called the Pixel Glitch Table (SCalGlitch). The initial version is released version 0. After a few months, with knowledge of the instrument behaviour, a new improved version (1) is released. Subsequent processing is expected to make use of the new version.

There may be one or more ***editions*** of a entity that are applicable for a given condition.

> For example, the SPIRE PixelMaskTable (SCalPixMask) calibration product is time-dependent, which means that one edition of that product will be applicable for one time range, and another edition for another time range.
> For example, PixelMaskTable edition A is valid for the period 1 Jan 2008 to 31 Dec 2009, while edition B is valid for the period 1 Jan 2010 to 31 Dec 2011. The applicable edition which should be used for an observation made during November 2009 is edition A.

Note that:

1. Each edition can have in turn a number of versions. The latest version would be the definitive one i.e. older versions are obsolete.
2. Each edition can have a new version at some point in time independent of the other editions.
3. Each version of each edition is a product.
4. Editions can be applicable to one or more of a number of possible parameters: time, flux, temperature, etc. In the most common case they will apply based on a single parameter: time, but the definition does not limit applicability to this case.

Functional Requirements

1. It shall be possible to store any number of versions of a product. *Example: calibration products, where all versions will be stored.*
2. It shall be possible to limit the number of versions of a product that are stored. *Example: pipeline standard products, where older versions are not systematically stored. It is possible that one older version might be stored for comparison purposes.*
3. It shall be possible to explicitly delete older versions. *Example: delete all versions except the last two.*
4. It shall be possible to retrieve the latest version of a product or edition. *It should not be necessary to retrieve all versions in order to do this. This should be the default behaviour in the absence of any explicit request. A version of an edition* is *a product, but is mentioned explicitly in order to stress the independence of editions from versions.*
5. It shall be possible to explicitly retrieve any given version of a product or edition. *It should not be necessary to retrieve all versions in order to do this.*

6. It shall be possible to retrieve all stored versions of a product or edition.
7. It shall be possible to retrieve the latest version of each product applicable to a specified edition. *The result is therefore a related set of data products. This requirement stresses the relation between editions. Note that a request for the* same *(e.g. first) version of each edition is largely meaningless and therefore not a requirement.*
8. It must not be possible to accidentally overwrite any version of a product. *Overwriting should therefore be explicit, if allowed at all*

Performance Requirements

1. These searches should take less then five seconds (less than one second is desirable) with a product pool containing 1000[1] products, on a laptop with 1GB memory and a 1.6GHz Centrino processor:
    a. Find a specified version of a given product.
    b. Find all latest versions of each edition of an entity. *In other words this requirement applies to finding all the related products as defined by editions.*
2. The memory requirement for retrieving a single version of a product must be less than double the size of the product. *This is another way of stating that it must not be necessary to load all versions into memory in order to find one of them.*

Design Requirements and implementation constraints

1. The functional behaviour must be identical for each supported product pool.
2. All products stored by the PAL shall have a defined version.
3. It shall be possible to deduce the version of a product from its contents. *In practice this probably means that versioning information is stored in the metadata. It is accepted that products are mutable and therefore theoretically possible that this information could be deleted, spoofed, or otherwise modified by sufficiently determined users.*
4. It shall be possible to determine whether any two products from any source are the same version of the same product. *As an example, this means that if the same version of a product is written to multiple different pools, it can be subsequently recognised that each of these products are the same version.*

[1] Total: each version of each edition counts as one.