PDS4 CHALLENGES IN THE PSA. J. Saiz[1], I. Barbarisi[1], R. Docasal[1], C. Rios[1], A. Montero[1], A. Macfarlane[1], C. Laantee[1], S. Besse[1], C. Vallat[1], J. Marcos[1], J. Arenas[1], J. Osinde[1], C. Arviset[1], [1]ESA/ESAC, Camino Bajo del Castillo s/n, Urb. Villafranca del Castillo, 28691 Villanueva de la Cañada, Spain, jsaiz@sciops.esa.int.

Introduction: The Planetary Science Archive (PSA) [1] is the central repository where products from all planetary missions of the European Space Agency (ESA) are stored, following the standards given by the Planetary Data System (PDS).

While legacy missions such as Giotto, Huygens, Venus Express and SMART-1, the Rosetta mission, currently in post-operations phase, and the still operational Mars Express, use the former PDS3 standard, newer missions like ExoMars 2016, ExoMars RSP, BepiColombo and Juice use or will use PDS4.

Design challenges: Adopting PDS4 as the standard for new missions while being compatible with previously existing PDS3 products in the same archive has driven a design with several difficulties to overcome:

Common data model. The ESA planetary archive maps the key metadata from PDS3 and PDS4 into a common data model [2] with the intention of providing transparency to the available data lookup services: the main web portal from where products can be searched, viewed and downloaded, the machine access interfaces supporting the Planetary Data Access Protocol (PDAP) [3] and the EuroPlanet-Table Access Protocol (EPN-TAP) [4], as well as the FTP browser.

This common mapping is a result of a thorough analysis of the commonalities shared by both standards. Although fairly different in terms and format, their basic concepts are similar. An effort has been made to extract them into common categories (Figure 1).

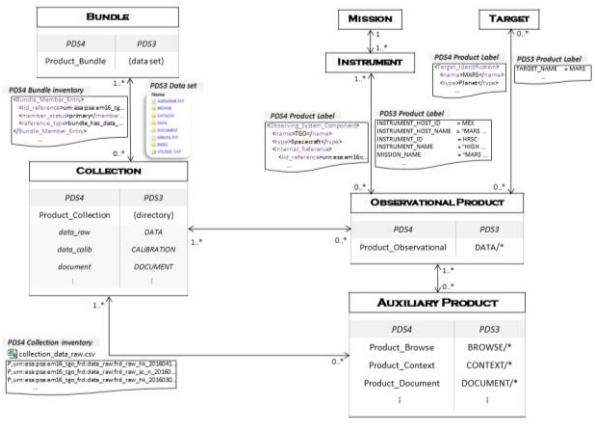

Figure 1. *PDS3 and PDS4 Data mapping*

Bundle generator. PDS4 products are organised at the top level into bundles [5]. Thus every delivery to the PSA coming from a PDS4 mission is expected to provide products within a bundle. Now, bundles have a non trivial structure to work with.

In order to make it easier for data providers to request products for being ingested into the PSA, it is permitted to deliver isolated products to the archive. Then it is the PSA ingestion software that is responsible for creating the required bundle for its proper ingestion (Figure 2).

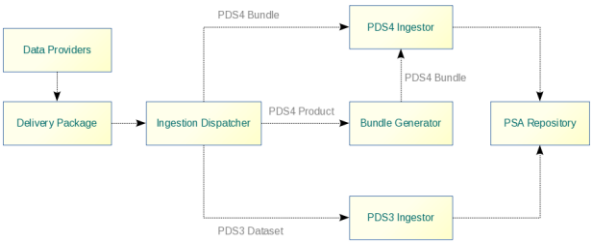

Figure 2. *Schematic view of the ingestion process in the PSA*

Creating a bundle and its delivery file, however, is not a straightforward operation, even for an automatic process, as the following paragraphs describe.

Validation procedure. Delivery files are expected to contain a transfer manifest, a checksum manifest and a label file, because in PDS4 every data item, including a delivery file, is considered a product, and each product has to be accompanied by a label file. All these have to be either validated, if a delivery file is passed to the PSA, or created for the generated bundle, if we talk about a single product delivery.

Subsequently, the data contents have to be validated too. This implies at least a syntax validation against the corresponding schema version, but also consistency checks to ensure that entities declared in the label files match the actual delivered contents, as well as verifying that values comply with *schematron* files.

The PSA will also delegate part of the validation process to software tools provided by the PDS that are yet to be incorporated.

Versioning of bundles, collections and products. Every time a given bundle is (re)delivered to the PSA, its version might be incremented. Bundle versioning is managed by the PSA so that it is kept consistent within the archive. The agreed convention with ESA missions employing PDS4 is to increase the minor version every week, and the major version every month. Therefore, when a bundle is going to be ingested, it is required for the ingestion process to look up and eventually retrieve its previous versions for a proper processing.

Deleting products considered erroneous or invalid by the data provider and reingesting them is a feature of the PSA that only applies to products that are not yet public. This task requires handling bundle and product versions with care in order not to create an undesirable maintenance problem.

Information Model (IM) versioning. Besides the aforementioned difference between PDS3 and PDS4 that the PSA has to manage, this latter standard has its own evolution. Roughly every six months, a new version of the PDS4 standard is released by the PDS. This implies a number of decisions to confront, both by the missions and by the archive:

- Which PDS4 version do we use?
- How often do we migrate to the last version?
- Do we upgrade all existing products?
- If each mission decides to use a different PDS4 version, how do we manage them in the PSA?

To facilitate this decision making to data providers, the PSA is building a flexible layout where various PDS4 versions are supported simultaneously: not only diverse information model versions between missions, but also different versions along the lifetime of a single mission. For this purpose, version independent Java interfaces have been created, which are implemented by adapter classes that make the appropriate translations to the corresponding schemas, with a minimal overhead. Though every time a new IM version is required to be incorporated, this common API could be slightly modified, the impact on client code is much less drastic than by exposing the generated JAXB [6] classes directly.

Ingestion time. As explained in previous paragraphs, creating a bundle involves many aspects, which may lead to a heavy and potentially slow processing. Consequently, it has been found that delivering fewer bundles with more products results in a more efficient strategy than trying to ingest numerous bundles with less products each.

Transactional ingestion. Once validated, adding a product or a bundle to the PSA comprises two clearly separated steps: ingestion in the data model (database), and importing the PDS4 files into the file repository. If either of the two fails, the whole operation has to be discarded, which has its own difficulty because, while transactions are nowadays well supported by database vendors, modifying and restoring the file system atomically is not so well provided by existing libraries.

PDS4 Updater. The received and imported PDS4 constructs contain all the information that providers have put in their delivery, but the data model reflects only part of them, a subset that is considered enough for the PSA interfaces to be exposed to users for searches and visualization purposes (the backed PDS4 data can be downloaded too, if necessary). This approach may lead to a situation in which the metadata stored in the database needs to be updated from the products of the repository, when for example a new key or some derived value is added to the data model and has to be filled from the existing imported files.

The alternative of reingesting these products would somewhat be an overreaction to this need. Therefore, it is foreseen to develop this functionality into the PSA to make it more tolerant to such update requests.

References: [1] Besse, S. et al., (2017) *Planetary and Space Science*; [2] Macfarlane, A. et al., (2017) *Planetary and Space Science*; [3] Salgado, J. et al. (2013), *IPDA Planetary Access Protocol*; [4] Erard, S. et al. (2014) *The EPN-TAP protocol for the Planetary Science Virtual Observatory*; [5] Data Design Working Group (2015) *PDS4 Concepts*; [6] E. Ort and B. Mehta (2003) *Java Architecture for XML Binding (JAXB)*.