The PDS4 Data Provider's Handbook

Guide to Archiving Planetary Data Using the PDS4 Standard

Version 1.7.0
April 2017

Contents

Tables and Figures

1 Introduction to the Data Provider's Handbook

The NASA Planetary Data System (PDS) has been archiving and distributing planetary science data since the 1980s. Starting in 2013 PDS began archiving data using a new standard known as Planetary Data System Version 4 (PDS4).

PDS4 represents a departure from previous versions of the PDS. It has been designed using contemporary information technology concepts and tools that were not available for earlier PDS versions. The system is built around a data model that rigorously defines each of its components and the relationships among them. There are only four fundamental data structures, but many extensions are possible — each rigorously defined. By carefully controlling product definitions and relationships, PDS can accurately track the progress of each product entering the system, compute detailed inventories of holdings, design sophisticated services that users can request to act on subsets of the archive (such as transformations and displays, in addition to the expected search and retrieval functions), and connect data products to relevant internal and external documentation.

The basis of PDS4 is the *Information Model* (IM) [1]. This is a set of defined concepts, objects, relationships, rules, and operations that represent the PDS archives. The model drives the development of PDS4 documentation and tools. This version of the *Data Provider's Handbook* (*DPH*) conforms to version 1.7.0 of the *PDS4 Information Model*. The IM is revised as the needs of data providers and users evolve, usually no more frequently than twice a year. As the IM is revised, this handbook will be revised as needed.

1.1 Purpose

The *DPH* is a guide for the preparation of data being submitted to PDS. As a data provider for a mission instrument team or an individual project, you are probably required to submit your science data to PDS. You should be assigned one of the PDS discipline nodes (Atmospheres, Cartography and Imaging Sciences, Geosciences, Navigation and Ancillary Information Facility, Planetary Plasma Interactions, Ring-Moon Systems, or Small Bodies) to work with you to prepare your data submission. The *DPH* should be used in conjunction with advice from staff at your *consulting node*. They will walk you through the preparation of products, collections of products, and bundles of collections, which are the units in which deliveries are made to PDS4.

1.2 Audience

The *DPH* is written for scientists and engineers in the planetary science community who are planning to submit new or restored data to PDS4 (data providers)[1]. The document is applicable to all such submissions, whether from mission instrument teams or individual data providers.

When the text directly addresses the reader as "you", it is referring to the data provider. When it says "we", it means the authors or PDS personnel.

[1] PDS4 standards are, in general, not backward-compatible with version 3 (PDS3). Some PDS3 structures are no longer supported under PDS4.

1.3 Reader Preparation

This handbook addresses all aspects of creating a PDS archive from defining the archive contents to delivery of the archive to PDS. The topics are presented in the sequence in which they are likely to occur during archiving preparation, and the discussions go from the general to the specific, with very detailed instructions removed to appendices so as not to disrupt the flow of the text. In Section 2 you will be introduced to the components that make up a PDS archive. Section 3 provides a checklist of archive preparation tasks, and the sections that follow address each item in the checklist at length. Appendix A spells out the many acronyms and abbreviations used in the handbook. Appendix B gives some observations and advice from other data providers who have experienced the development of PDS archives from beginning to end. The other appendices give detailed instructions for particular tasks, and they are mentioned in the main text where relevant.

The *DPH* is associated with a set of example archives online at https://pds.nasa.gov/pds4/doc/examples. The text will often refer to these examples, and excerpts from the examples are reproduced in the text to illustrate the discussion.

The instructions in this handbook mention PDS requirements and PDS recommendations. When a requirement is mentioned, a reference is given to one of the controlling documents (see Section 1.4.1 below) where the requirement is fully stated. When a recommendation is given, it is based on the accumulated experience and recognized best practices of PDS personnel. If a statement in this handbook is found to conflict with a statement in one of the controlling documents, the controlling document will prevail. If you are in doubt about the best course to take, consult your PDS discipline node.

All readers, even those very familiar with previous versions of PDS, should read the *PDS4 Concepts* document [5, below] before beginning the *Data Provider's Handbook.* The *Concepts* document also includes a glossary of PDS-related terms that you may wish to keep handy.

1.4 Applicable Documents

The *DPH* is one of several documents that describe the PDS4 system. Some of these documents are considered to represent the official PDS4 Standards; these are listed in Section 1.4.1 below. Other documents are provided to help readers understand and use PDS4. The *DPH* is in the latter category. Throughout this handbook you will find references to these documents by their title and number in the list, e.g. *PDS Standards Reference* [2].

The *DPH* should be used in conjunction with the *PDS Standards Reference* [2], the *PDS4 Data Dictionary* [3], and the *PDS4 Common XML Schema and PDS4 Schematron*[2] [4], which collectively provide the information necessary to develop a PDS4-compliant archive.

[2] The PDS4 Schema and Schematron files are software-readable representations of the PDS4 Information Model. They are discussed in Section 6.1.

1.4.1 PDS4 Controlling Documents

The following documents are derived directly from the PDS4 Information Model and collectively represent the PDS4 Standards.

1. *Planetary Data System (PDS) PDS4 Information Model Specification*, Version 1.7.0.0, https://pds.nasa.gov/pds4/doc/im/current/, September 28, 2016.
2. *Planetary Data System Standards Reference*, Version 1.7.0, https://pds.nasa.gov/pds4/doc/sr/current/, September 15, 2016.
3. *PDS4 Data Dictionary, Abridged*, Version 1.7.0.0, https://pds.nasa.gov/pds4/doc/dd/current/, September 28, 2016.
4. *PDS4 Common XML Schema and PDS4 Schematron*, Version 1.7.0.0, September 30, 2016, and other Schemas and Schematron files recognized in PDS4, https://pds.nasa.gov/pds4/schema/released/.

1.4.2 Other PDS4 Documents

5. *PDS4 Concepts*, Version 1.7.0, https://pds.nasa.gov/pds4/doc/concepts/, September 15, 2016. This document provides a high-level overview of PDS4, and should be the first document read by someone new to PDS4.
6. *PDS4 Data Provider's Handbook*, Version 1.7.0, https://pds.nasa.gov/pds4/doc/dph/current/, April 1, 2017. This is the document you are currently reading.
7. *PDS4 Data Provider's Examples*, Version 1.7.0.0, https://pds.nasa.gov/pds4/doc/examples, October 1, 2016. The examples are sets of products, collections, bundles, and packages that illustrate the use of PDS4.
8. *Ingest LDD Users Guide*, Version 1.2.1, November 4, 2015. This document is included in the LDDTool software package on the PDS4 Software web site, https://pds.nasa.gov/pds4/software/ldd/.

In general all PDS4 documents can be found online at https://pds.nasa.gov/pds4/doc. For questions concerning these documents, consult your PDS discipline node or contact the PDS Operator at pds_operator@jpl.nasa.gov or 818-393-7165.

1.4.3 PDS4 Software

PDS provides software tools to aid in generating, validating, and transforming PDS4 data products and labels. These tools are online at https://pds.nasa.gov/pds4/software/.

PDS also maintains a Tool Registry in which data providers and users may share tools they have created for use with PDS4 data. Submissions to the Tool Registry are vetted by the PDS Engineering Node. The Tool Registry is online at https://pds.nasa.gov/tools/tool-registry/.

1.4.4 PDS Small Bodies Node Resources

The PDS Small Bodies Node maintains a set of web pages and a Wiki site to help explain PDS4 to users who are unfamiliar with it. The material is for PDS4 archive developers both inside and

outside of the Small Bodies Node. The web pages begin at http://pdssbn.astro.umd.edu/howto/understand_PDS4.shtml.

The wiki is available at http://sbndev.astro.umd.edu/wiki/SBN_PDS4_Wiki.

2 Overview of a PDS4 Archive

2.1 Components of a PDS4 Archive

PDS holdings include many archives of planetary data going back to the early years of space exploration in the 1960s and 1970s. Often you will hear the whole of PDS holdings called the PDS Archive, but usually PDS users speak of *an archive*, meaning a set of data products, documentation, and related material created by a data provider. PDS4 introduced the specific terms *bundle* and *collection*, which are explained below, but the term *archive* is still used. The important point is that a PDS archive contains more than just science data. It includes descriptions of each product — sufficient documentation to enable a user who is unfamiliar with the archive to read, understand, and use the data — and often other kinds of products that may help the user, such as calibration data or browse data (quick-look versions of data to make searching easier).

This section describes the components of an archive and defines some PDS4-specific terms.

2.2 Products

Products are, of course, the heart of an archive. Before diving into what products contain — the science data, calibrations, documentation, etc. — we need to spend a moment on terminology. A product is the combination of one or more data *objects* and their associated *descriptions objects*; the descriptions are sometimes called *metadata* (data about data), which are concatenated into a PDS *label*, a separate ASCII file. A data object and its associated description object is sometimes called an *information object*.

There are three kinds of data objects. *Digital objec*ts are the things we find in PDS archives that carry the scientific measurements — tables, images, spectra, etc. No digital object may extend beyond a single file and no two digital objects may overlap within a single file. But multiple digital objects can be stored in separate files. *Physical objects* are things that are important to planetary exploration but that we can't put into the archive because they aren't digital — planets, satellites, telescopes, spacecraft, etc. *Conceptual objects* also don't fit in the archive; they are ideas, organizations, plans, etc., which don't exist in a physical sense but are still important. Note that an atlas of Mercury or plans to explore asteroids may be printed on paper or stored in computer files — the latter are digital objects; but Mercury and the exploration plans themselves are physical and conceptual objects, respectively. Even though we won't find physical and conceptual objects in PDS, we can still create descriptions (labels) for them; the resulting products (actually the labels only) can be included in PDS archives.

2.2.1 Science Data Products

Science digital objects are generated by instruments on spacecraft visiting solar system bodies or in laboratories or observatories on Earth. They may be wrapped with other digital objects, labeled, and delivered to PDS by mission instrument teams or by individual investigators. They range in processing level from the raw measurements recorded by an instrument to highly derived products such as topographic maps.

PDS recognizes five broad levels for categorizing science data processing; see the PDS Policy on Data Processing Levels on the *PDS Policies* web page (https://pds.nasa.gov/policy/). These are shown in Table 2-1.

Table 2-1. Science Data Processing Levels

Processing Level	Definition
Telemetry	An encoded byte stream used to transfer data from one or more instruments to temporary storage where the raw instrument data will be extracted. **PDS does not archive telemetry data.**
Raw	Original data from an instrument. If compression, reformatting, packetization, or other translation has been applied to facilitate data transmission or storage, those processes will be reversed so that the archived data are in a PDS-approved archive format.
Partially Processed	Data that have been processed beyond the raw stage but which have not yet reached calibrated status.
Calibrated	Data converted to physical units, which makes values independent of the instrument.
Derived	Results that have been distilled from one or more calibrated data products (for example, maps, gravity or magnetic fields, or ring particle size distributions). Supplementary data used to interpret observational data, such as calibration tables or tables of viewing geometry, should also be classified as derived data if not easily matched to one of the other categories.

Data providers may find this breakdown useful when deciding what archives to submit to PDS. It is not necessary to archive all processing levels from raw to derived; however, based on requests from the science community, PDS encourages providers of raw data to submit calibrated data also. The archive requirements for metadata and documentation are the same regardless of processing level.

2.2.2 Other Kinds of Products

PDS4 is designed using the principle "Everything is a product." For example, a table of science measurements is a product once it has been labeled (per the previous section). A document that describes the experiment is also a product (with its label). An image showing a plot of the measurements (and its label) is a product. Examples of products other than science data products are calibration data, calibration reports, browse images, user's guides, published papers (with the journal's permission), and any other kind of documentation that is needed to use the science data.

2.3 Metadata in PDS4 Labels

Metadata is defined simply as "data about data". A product – in particular, a science data product – is useless without accompanying information about its content and structure. Metadata about

content may include, for instance, the time and location of a science observation, the instrument that made the observation, and the conditions of the observation. Metadata about structure includes the information needed to read the digital object, such as the dimensions and data type of an image, or the description, size, and data types of columns in a table.

PDS4 labels are written in eXtensible Markup Language (XML). XML was chosen because it is a language that can be read both by humans and by software and because it is a widely used international standard for which a large amount of software already exists. We will discuss XML labels in more detail in Section 6 and in Appendix D and Appendix E.

To see examples of XML labels for PDS4 products, visit https://pds.nasa.gov/pds4/doc/examples/, choose the latest set of PDS4 Example Products, and look through the subdirectories for files with the .xml extension.

2.4 Products, Collections, and Bundles

Section 8 of *Concepts* [5] is a good introduction to the topic of products, collections, and bundles. Section 2A of the *Standards Reference* [2] develops the topic further. To summarize, products that are related in some way may be grouped together in a collection, and, similarly, related collections may be grouped together into a bundle. These are the three levels of data organization in a PDS archive. Typically, a bundle is what you deliver to PDS.[3]

For example, data providers for a planetary mission could decide that each instrument team will deliver data from its instrument in a separate bundle. An instrument team could decide that its bundle will consist of a separate collection for each level of science data products – raw, calibrated, and derived – along with a document collection and perhaps a collection of browse products. These choices are made in consultation with the instrument team's assigned PDS discipline node, and with the PDS node that is assigned to be the lead node for the whole mission.

An individual data provider will probably have a simpler arrangement, delivering a single bundle consisting of one or more data collections and a document collection. Again, the decision is made in consultation with your PDS node.

To see an example of a PDS4 bundle with a typical set of collections, visit https://pds.nasa.gov/pds4/doc/examples/, and follow the latest link to a PDS4 Example Archive.

2.5 Unique Identifiers

Every product in a PDS4 archive has a unique Logical Identifier (LID). It also has a Version Identifier (VID). These are often used together as a LIDVID. When a product is revised, its LID remains the same but its VID is incremented. A LIDVID is guaranteed to be unique across the whole PDS system. To ensure uniqueness, when you create LIDVIDs for your products you must follow the rules for forming a LIDVID as specified in the *Standards Reference* [2] Section 6D. We will discuss this in detail later in Section 5 of this handbook. For now, an example will illustrate the concept.

[3] For archives that accumulate over time, such as for a planetary mission, a typical delivery to PDS consists of a subset of a bundle, perhaps only a collection or part of a collection.

LIDVID formation rule:
`urn:nasa:pds:<bundle_id>:<collection_id>:<product_id>::<version_id>`

Example:
`urn:nasa:pds:bopps2014:calibrated:ceha_1_024109424_n011_0244e_fit::1.0`

The example LIDVID is for version `1.0` of the data product `ceha_1_024109424_n011_0244e_fit` in the `calibrated` collection of the `bopps2014` bundle. The prefix `urn:nasa:pds` is required for all PDS LIDVIDs. (*Standards Reference* [2], Section 6D.2.)

As mentioned above, everything in PDS4 is a product. Collections and bundles are also considered products, and they also have unique LIDVIDs. In the above example, the bundle LIDVID is `urn:nasa:pds:bopps2014::1.0` and the collection LIDVID is `urn:nasa:pds:bopps2014:calibrated::1.0`.

In this document the term *basic product* refers to all types of products except collection and bundle products.

2.6 Context Products

Context products are a special category of PDS4 products. They provide a way to associate the digital material in an archive with a physical or conceptual object that is outside the archive. Context products are described in Section 8B of the *Standards Reference* [2], so we'll only summarize here.

Context products exist for physical and conceptual objects relevant to a product, collection, or bundle — mission, spacecraft, instrument, laboratory, observatory, telescope, planetary body, etc. Context products are created by data providers, but they are finalized and managed in a PDS master context bundle so that all data providers can access and use the same material.

Because the context product references a physical or conceptual object, it consists of a label only. The label contains information that identifies and very briefly describes the physical or conceptual object; it also includes references to published documents for more complete information. Section 4.1 below tells more about the different kinds of objects covered. Section 9 goes into detail about creating and using context products. If a context product already exists for your data object, you should review its content for scope and accuracy; if it is sufficient, you do not need to do anything more.

Like all products, context products have LIDVIDs. If your archive will require the creation of new context products (or upgrading of existing context products), you will need to work with your consulting node and the PDS Engineering Node (EN) to choose the right LIDVIDs for them. The PDS Engineering Node is the steward of context products; EN staff approve LIDs for these products. Appendix L is a guide to forming LIDs for context products.

To see examples of context products, visit https://pds.nasa.gov/pds4/doc/examples/, follow the latest link to a PDS4 Example Archive, and look in the context directory.

3 Checklist for Creating a PDS4 Archive

The following checklist is provided to help you plan the development of your archive. The steps are given in the order that they should be accomplished. Each item is discussed fully in the following sections.

1. Define archive contents: data products, documentation, and any additional material (Section 4).
2. Define archive organization: collections, bundles, directory structure (Section 4).
3. Assign unique LIDs to collections and bundles; determine LID scheme and file naming scheme for data products (Section 5).
4. Design labels for data products, documents, and any additional material (Section 6).
5. Develop procedures for generating many labels systematically (Section 7).
6. Generate products and labels (Section 7).
7. Generate documentation for the archive (Section 8).
8. Create context products if they do not already exist (Section 9).
9. Assemble archive for peer review (Section 10).
10. Validate archive (Section 11).
11. Submit archive to peer review (Section 12).
12. Submit final archive to PDS according to delivery schedule (Section 13).

Before getting started, you may find it helpful to read Appendix B, Archive Lessons Learned from Past Missions.

4 Defining Archive Contents and Organization

4.1 Terminology

As defined in Section 2.2, PDS recognizes three kinds of data objects — digital, physical, and conceptual. Each of the three can be combined with a description object to form an information object, and one or more information objects may then be wrapped into a product. Of the three types of data objects, only digital objects will accompany their descriptions into an archive (dust samples, spacecraft, missions, space agencies, etc. cannot be stored in computers). If many data objects have similar characteristics, we can group them into a *class*, and the common characteristics are the defining *attributes* of that class. Classes and attributes are primary building blocks of PDS labels.

A product is one or more closely related data objects for which the descriptions have been combined into a single XML label and for which there is a PDS-unique logical identifier. Closely related products may be grouped into a collection; in fact, every product entering PDS must be a member of some collection. Closely related collections may be grouped into a bundle. Every collection in PDS must belong to a bundle. [Reference]

Commented [EAG1]: It is somewhat implied in section 2 of SR, but not stated.

For example, a planetary image, the histogram of its pixel values, and the descriptions of both could be organized as a product. Many such products — perhaps of the same target — could be defined as a collection. Image collections from many targets along with appropriate documentation collections and calibration collections could be a bundle, which would be a deliverable to PDS.

A few words have meanings that differ depending on the community in which they are used. We have adopted modifiers to help distinguish among multiple uses. For example, *attribute* is widely used in both PDS and XML — but its meaning in each case is different. In this document we use *attribute* (for PDS) and *XML attribute* to establish the context.

Warning: We have avoided using terms that have strong PDS3 connotations when the PDS4 meanings are different. Unfortunately, the English language does not provide a sufficient set of meaningful, unique, unambiguous terms to meet all of our needs. Please do not infer meanings from past experiences – review the PDS4 definitions in the PDS4 Glossary, Appendix A in the PDS4 *Concepts* document [5].

4.2 Archive Design

In consultation with your PDS discipline node (DN), begin by reviewing the products you expect to include in your archive. Group related products into collections and related collections into bundles. A very simple archive could fit into a single bundle with half a dozen collections, each holding a handful of products. In some cases you might want to collaborate with others and jointly produce one bundle, where your data would be in one of several contributed collections. A large archive could have many bundles and hundreds of data collections.

You will likely have at least one data collection and probably one document collection. You may also want to include other material such as browse products, which are reduced-size images of data products that a user may scan quickly to help decide what data are of interest. Calibration data and reports are required by some missions. PDS does not require any particular collections

to be in a bundle, but your submission must be in the form of at least one bundle with at least one collection.

For science data products, consider how you want to aggregate the measurements from your instrument. Should one product consist of a day's worth of data, or an hour's worth, or should it be grouped by some other parameter besides time? If the archive is for a mission, you may need to consider how data products from other instruments on the mission are aggregated so that yours will fit in easily.

What processing levels of science data do you want to archive? Refer back to Table 2-1. If you are archiving multiple levels of data, you may wish to put each one in a separate collection. There is no PDS requirement for archiving a particular processing level, but there may be mission requirements. Most mission data providers want to archive at least the raw products to preserve them, but most users need the calibrated and derived products, especially if the processing is complicated. Keep in mind that your archive will be peer reviewed, and that reviewers may insist on your providing higher-level products (Section 12).

One important question that should be decided at this point is the form that your data products will take. The next section addresses this topic in detail.

4.2.1 Data Formats

Four basic structural data formats are allowed in PDS4.

1. Arrays
   - Suitable for raster images with two or more (up to 16) dimensions.
   - The elements of an array are homogeneous – all elements have the same data type, and only binary data types are allowed.
   - The individual elements of any array are stored with their bytes in the order dictated by their scalar type — that is, least- or most-significant byte first (your choice, but you must specify).
   - The array elements are stored in the axis order described in *Standards Reference* [2], Section 4A — that is, last index fastest, or row-major order for 2D arrays.
2. Repeating record structure
   - Suitable for tables with fixed-width columns.
   - May be either binary or character, but not both.
   - The fields of a record may be heterogeneous – they may have different data types within the binary or character choice above.
   - Any single field must be homogeneous from one record to another.

The majority of PDS4 data can be supported by these two structures. For those PDS4 objects which cannot be supported by the above, there are two additional structures distinguished by whether or not software must be used to decode the information before it can be accessed for reading, display, or analysis.

3. Parsable structure

- Suitable for plain text, and tables with variable-length fields and records (delimited text such as comma-separated value (CSV) format).
- The contents are a byte stream which can be parsed with standard rules (e.g., comma separated entries, standard punctuation); no decoding software is required (e.g., Adobe Acrobat©).
- The attribute **parsing_standard_id** is used to identify the parsing standard to be used. Examples are "7-Bit ASCII Text" and "UTF-8 Text". See Table 4-1 for other values under subclasses of Parsable_Byte_Stream.

Table 4-1. Values for parsing _standard_id

Parsable_Byte_Stream subclass	parsing_standard_id value
[tbd by RJ]	

4. Encoded structure
   - Suitable for complex documents, browse products, etc., but generally not for science data products.
   - Contents are a byte stream that must be decoded by software before use.
   - The use of encoded structure objects is restricted by PDS to a limited set of PDS approved external standards (e.g., PDF/A, JPEG, and GIF).
   - Only in exceptional cases will encoded structure objects be considered appropriate for storing observational – that is, science – data. Prior PDS approval is required.
   - The attribute **encoding_standard_id** is used to identify the encoding standard to be used. Examples are "PDF/A", "GIF", "JPEG", and "J2C". See Table 4-2 for other values under subclasses of Encoded_Byte_Stream.

Table 4-2. Values for encoding_standard_id

Encoded_Byte_Stream subclass	encoding_standard_id value
[tbd by RJ]	

The *PDS4 Information Model* [1] defines a base class for each of these four structures: **Array**, **Table_Base**, **Parsable_Byte_Stream**, and **Encoded_Byte_Stream**. A base class is a generic definition that can be built upon to form more specific definitions, without losing the basic features of the class. For instance, the **Array** base class is used to form the more specific **Array_2D** class and from that the **Array_2D_Image** class.

4.2.2 PDS Policy on Formats for Science Data, Supplemental Data, and Documentation

The above four structures cover all types of products in a PDS archive. There are further restrictions on science data products (also called observational products). The general philosophy is that in order to preserve data for the long term, formats must be as simple as possible, well-described, and not reliant on specific software, because that software may be unavailable in the future. PDS recognizes that the goal of preservation is often in conflict with another PDS goal, which is to make the data easily available to the science community today. To meet this need PDS allows copies of data to be archived in supplemental formats. PDS also provides tools that transform products from archive-quality formats to various other formats currently popular with users.

The PDS Management Council has adopted a *Policy on Formats for Data and Documentation*, and the list of supplemental formats is available online on the PDS Policies page, https://pds.nasa.gov/policy. The PDS Transform Tool is available on the PDS4 Software page, https://pds.nasa.gov/pds4/software.

Here is a summary of the policies. Refer to the policy page above for details.

Science data must be archived in PDS using only these formats:

- Binary or ASCII tables with fixed-width, identically structured records
- ASCII tables with delimited records, such as CSV files
- Binary arrays of no more than 16 dimensions
- SPICE kernels[4].

Documentation must be archived using only these formats:

- UTF-8 text without markup (e.g., no HTML). UTF-8 includes traditional 7-bit ASCII text.
- PDF/A-1a (preferred) or PDF/A-1b (see http://www.digitalpreservation.gov/formats/fdd/fdd000251.shtml).

Figures that accompany documentation may be embedded in a PDF document or provided as separate files formatted as JPEG, GIF, PNG, or TIFF images.

4.2.3 Data Storage Considerations

Here are a few more rules to consider when designing your data products. For a full description, see the *Standards Reference* [2], Section 4.

[4] SPICE kernels use a special format defined by the PDS NAIF (Navigation and Ancillary Information) Node for storing navigation and geometry data (http://naif.jpl.nasa.gov).

- Each digital object must be stored in one of the four basic data formats.
- A digital object must be contained in a single file; i.e., a digital object cannot span multiple files.
- A file may contain multiple digital objects[5].
- Multiple digital objects within a file are not required to use the same storage structure.
- When multiple digital objects are contained in a single file, they must be contiguous; they may not overlap in storage.
- Binary data in tables or arrays may be stored as 1, 2, 4, or 8-byte integers, signed or unsigned, or as 4 or 8-byte floating-point numbers in IEEE 754 format, or as complex numbers formed from floating-point components. Bytes may be stored in most-significant-byte-first order (MSB, "big-endian") or in least-significant-byte-first order (LSB, "little-endian").

4.2.4 Example Data

This document frequently refers to the set of PDS4 examples online at https://pds.nasa.gov/pds4/doc/examples/. On this page you will find a set of examples for each major release of PDS4. When designing a new data product it is best to use the most recently released examples.

There are two different sets of examples:

- A set of example products. This includes a representative set of products that would exist within an archive (e.g., character table, binary table, document, etc.).
- An example of a complete archive — a bundle with collections, each having products.

4.3 Organizing the Bundle and Collections

The next step in designing an archive is to organize the data products into collections and the collections into bundles.

For simplicity, imagine that your archive includes a single table of observational data (and its label), and that the remainder of the archive consists of supplementary information, which will help future scientists understand and use the data. We will organize this material into a single bundle with three collections:

Browse Collection

Data Collection

Document Collection

The data, obviously, will be in the data collection; an abbreviated or reduced-resolution version of the data will be in the browse collection. Documents will go into the document collection.

[5] Except for documents, where there is a limit of one object per file.

These are typical collections, but none of these collections is required. See *Standards Reference* [2] Section 2A.5 for other types of collections that may be used. It's best to consult your PDS node about what collections to include.

4.3.1 Directory Organization

PDS does not set requirements for the physical storage organization of an archive (see *Standards Reference* [2], Section 2B.1), but you will probably find it helpful to let the physical organization of directories follow the logical organization of bundles and collections. Your bundle can be organized into a simple directory structure with one directory in the bundle root for each collection, as shown below. A file whose name ends in .xml is a label that describes the file with the same name but a different extension; for example, **collection_data.xml** and **collection_data.csv**. In some cases we have simplified the structure by showing products rather than data file/label pairs.

```
bundle root
| - bundle.xml
| - readme.txt
|
| - browse
|     | - collection_browse.xml
|     | - collection_browse.csv
|     |
|     | - browse_ product
|
|
| - data
|     | - collection_data.xml
|     | - collection_data.csv
|     |
|     | - data_file.tab
|     | - data_file.xml
|
| - document
      | - collection_document.xml
      | - collection_document.csv
      |
      | - errata_document_product
      | - mission_document_product
      | - instrument_document_product
```

The root level subdirectories each correspond to a single collection and have the name of the collection. Each subdirectory contains a collection inventory file and its XML label file.

The bundle root contains **bundle.xml**, the XML label file for the bundle product, and **readme.txt**, an optional file that is described in **bundle.xml**.

4.3.2 Directory and File Naming

There is a one-to-one correspondence between the collections in your bundle and the root level subdirectories. Beneath this level you may create subdirectories as needed.

In the example above, there is only one observational data file in the data directory, **data_file.tab**, so no data subdirectories are needed. If there are more than about 20 observational data files, you will need to establish subdirectories under the data directory. Choose a directory organization that seems natural for your data, balancing the number of files in a subdirectory and the number of subdirectories in a path. Grouping the data products by time is a common method — for example, having directory names based on the UTC date, each directory containing files with names based on the UTC time (*Standards Reference* [2], Section 2B.2.2.2).

It is common practice to use the label file name as the last component of the Logical Identifier (LID) for the data product, so if you intend to do this, you should choose a file naming scheme for your data products that results in a unique name for each file, even files that are in separate subdirectories. It is tempting to put as much identifying information as you can fit into the file name, for quick selection of products without having to open up the labels. This can result in long and cumbersome file names that are hard to read, so aim for a balanced approach. You can use dashes and underscores to separate parts of a file name for readability.

Rules for forming file and directory names are given in *Standards Reference* [2], Section 6C. Here are a few things to keep in mind.

- The directory name must be unique within its parent directory.
- The file name must be unique within its parent directory. The same file name may appear in different directories.
- Directory and file names must be no longer than 255 characters. Some operating systems limit the length of path names, so we recommend keeping the total path length under 255.
- Directory and file names must be case-insensitive; for example, **MyFile.txt** and **myfile.txt** are not permitted in the same directory.
- Directory and file names must be constructed from the character set

A-Z	ASCII 0x41 through 0x5A,
a-z	ASCII 0x61 through 0x7A,
0-9	ASCII 0x30 through 0x39,
dash “-”	ASCII 0x2D,
underscore “_”	ASCII 0x5F, and
period “.”	ASCII 0x2E.

- Directory and file names must not begin or end with a dash, underscore, or period.
- The file name must include at least one period followed by an extension. A file name may have more than one period, but PDS will consider all periods other than the final one to be part of the base name.
- Using the same base name for label and data file names is considered good practice; but this is not a PDS4 requirement and is not always possible, such as when a product consists of two or more data files.

As is discussed in Appendix B of this document, if you are archiving data as part of a mission, it is often advantageous to negotiate these naming conventions well in advance, in the context of

the entire mission, and with regard to naming conventions adopted by past missions with similar instruments.

4.3.3 Determining the Documentation Needed

You and your consulting PDS node should sketch out the contents of the document collection early in the design process. PDS requires that products, collections, and bundles be documented so that scientists in future years can understand (1) how the data were collected and processed, (2) what the data mean, and (3) the limitations of the data. Refer to the *Standards Reference* [2], Section 8, and confer with your PDS representative to determine all of the required and appropriate documentation for the document collection.

Documentation takes three forms in PDS: (a) XML labels, (b) documents included within the archive, and (c) references to material that is publicly available elsewhere. The archive design should include a clear plan for how the documentation will be distributed among these three categories.

Documentation considered essential to understanding or using the archive, except for published journal articles, must be submitted as part of the archive. Journal articles may be included if permitted by the copyright holder; otherwise, they should be cited in the archive documentation.

As an example, documentation might include the following:

- An errata file that describes any changes, known errors, or anomalies in the archive.
- A copy of a published journal article that describes the mission.
- A copy of a published journal article that describes the instrument.
- A User's Guide that explains how to read and interpret the data.

For mission archives, it is common practice to include the Software Interface Specification (SIS) document that the mission may require the instrument team to generate. This document goes into the details of the structure, content, and processing of the data products. Missions may also require teams to generate calibration plans, data, and/or reports, which are also suitable for inclusion in the archive.

PDS-compliant formats for documents are UTF-8 and PDF/A. UTF-8 is plain text, including traditional 7-bit ASCII text files. In a PDS archive UTF-8 documents may not include any embedded markup language such as HTML. PDF/A is an ISO-standardized format of PDF (Portable Document Format) suitable for long-term archiving. PDF/A-1a, the preferred level of PDF/A, means a document is in complete compliance with the ISO requirements. PDF/A-1b, also acceptable in PDS, means a document is in minimal compliance. Many software applications that generate PDF files are capable of generating PDF/A files. Your PDS representative can convert standard PDF documents to PDF/A if needed.

Once you have a document in UTF-8 or PDF/A format, you may provide the same document in additional formats if you think they will be convenient. For example, if you have a PDF/A document, you may wish to provide a Microsoft Word version of it also. See the *PDS Policy on Formats for PDS4 Data and Documentation* at https://pds.nasa.gov/policy/format_policies_final.pdf, and the *List of Supplemental Formats* at https://pds.nasa.gov/policy/Supplemental_Formats.pdf.

5 Assigning Unique Identifiers

When you have determined the contents and organization of your archive, and you have settled on a directory and file naming scheme, the next step is to make a plan for uniquely identifying the products in the archive. As we mentioned in Section 2.5, Unique Identifiers, all products – data, documentation, and other components – carry a unique Logical Identifier (LID) that follows a set of formation rules. It's important to determine these LIDs before you go further in creating labels for your products because the LIDs are needed for the labels. Also, you and your PDS representative will need to work with a data engineer at the PDS Engineering Node to ensure that your LIDs are unique across PDS and, for mission data, to ensure that they conform to the mission-wide LID design.

5.1 General Concepts

Every product has an identifier that is unique across all products registered and archived with the PDS. This identifier is referred to as a LIDVID and is the concatenation of a logical identifier (LID) and a version identifier (VID). We'll address the construction of each in the following sections.

Here are some general rules:

- LIDs must be unique across PDS.
- One LID covers all versions of a single product; the VID distinguishes among versions[6]. Rules for incrementing VIDs are found in the *Standards Reference* [2] Section 6D.3.
- Each LID in a PDS archive begins with the agency identifier `urn:nasa:pds:`[7]. This ensures that any LID that is unique within PDS is also unique within NASA and within the global federation of agencies that subscribe to this identification system — that is, the prefix ensures that products have globally unique identifiers.
- LIDs are restricted to lower-case letters, digits, the dash, the period, and the underscore. Colons are also used but only in a prescribed way to delimit fields (see below).
- Each PDS4 LID is constructed as four (bundle), five (collection), or six (product) fields, where each pair of fields is separated by a colon. Each field must begin with a letter or digit.
- LID maximum length is 255 characters.

[6] Discuss with your consulting node the conditions under which a new version should actually be a different product. Versions generally succeed each other and result from improvements — incrementally better calibration, for example. If the processing algorithm itself changes significantly, you may want to consider defining a new set of products rather than incrementing the version.

[7] You may find other prefixes in archives maintained by other agencies (for example, urn:esa:psa: for the European Space Agency PSA archive), but do not use these when preparing data for delivery to PDS.

5.2 Constructing Logical Identifiers (LIDs)

The complete set of requirements for LID construction is given in Section 6D.2 of the *Standards Reference* [2]. We give a summary here. LIDs for context products have special requirements, which are discussed in Appendix L.

Recall that each basic product is delivered to PDS as a member of a collection and that each collection is a member of a bundle. LIDs are constructed based on a hierarchical set of these relationships.

LIDs are constructed by concatenating fields of characters. The fields are separated by colons. This is the only use of colons permitted in LIDs.

- Bundle LIDs are constructed by appending a unique bundle-specific identifier to the agency identifier, such as `urn:nasa:pds` or `urn:esa:psa`.

  Bundle LID = `urn:nasa:pds:<bundle_id>`

  Bundle LID = `urn:esa:psa:<bundle_id>`

  Since all PDS bundle LIDs are constructed this way, the bundle LID will be globally unique.

- Collection LIDs are constructed by appending a unique collection identifier to the parent bundle's LID, for example

  Collection LID = `urn:nasa:pds:<bundle_id>:<collection_id>`

  Since all PDS collection LIDs are constructed in this way and the collection identifier is unique within the bundle LID, the collection LID will be globally unique.

- Basic Product LIDs are constructed by appending a unique product identifier to the parent collection's LID.

  Product LID = `urn:nasa:pds:<bundle_id>:<collection_id>:<product_id>`

  Since the product LID is based on the collection LID, which is unique across PDS, the product LID will be globally unique.

5.2.1 Examples

The following examples are based on a hypothetical mission.

	Name	***Abbreviation***
spacecraft	Super SpaceCraft 01	ssc01
instrument	High Resolution Photon Counter	Hirespc
mission phase	Cruise, Mercury, Earth phase	Cruise

The Hirespc instrument team decides to use the spacecraft clock count (sclock) at the start of each observation as the product field of the LID for observational data products. They also

decide to create a separate bundle for each phase of the mission. This is all the information we need to start designing LIDs.

Cruise Phase Bundle

Bundle LID

`urn:nasa:pds:ssc01.hirespc.cruise`

Note that in the above example bundle field, ssc01.hirespc.cruise, we used periods as separators. Alternatively we could have used dashes, underscores, or some combination of the three. Discuss the use of period, dash, and underscore in LIDs with your consulting node; there may be a preference.

Collection LIDs

`urn:nasa:pds:ssc01.hirespc.cruise:browse`

`urn:nasa:pds:ssc01.hirespc.cruise:context`

`urn:nasa:pds:ssc01.hirespc.cruise:data`

`urn:nasa:pds:ssc01.hirespc.cruise:document`

`urn:nasa:pds:ssc01.hirespc.cruise:xml_schema`

If there is a large number of products, it might be desirable to subdivide the data products into two or more collections by processing level (data_raw, data_derived, etc.), by year (data_2006, data_2007, etc.), or by a different discriminator. Many discriminators are permitted; you should use whichever is best suited to your data. Discuss with your consulting node.

Product LIDs [examples of data products identified by a spacecraft clock count of 31234567 in various collections]

`urn:nasa:pds:ssc01.hirespc.cruise:browse:browse_31234567`

`urn:nasa:pds:ssc01.hirespc.cruise:data:data_raw_31234567`

`urn:nasa:pds:ssc01.hirespc.cruise:data:data_derived_31234567`

`urn:nasa:pds:ssc01.hirespc.cruise:document:errata`

5.3 Constructing Version Identifiers (VIDs)

Detailed requirements and formation rules for versioning LIDs are provided in the *Standards Reference* [2], Section 6D.3; we provide a brief summary here.

Version IDs are used for all types of products, including basic products, collections, and bundles.

- VIDs are appended to LIDs by a double colon ("::").
- VIDs must be of the form `M.n`, where `M` denotes a major version and `n` denotes a minor version.[8]

[8] The PDS4 Information Model, data dictionaries and related schemas use a four-field VID, M.n.o.p, in decreasing order of significance. See *Standards Reference* [2] Section 6D.3.2.

- The major number (M) is initialized to 1 for archive products. Whenever the major number (M) is incremented, the minor number (n) is reset to 0.
- Neither M nor n should be prepended with zeros; each is simply incremented as an integer. Thus 1.1 and 1.10 are different versions, and 1.01 is invalid.

5.4 LIDVID Construction

A *version identifier* (VID) may be appended to a logical identifier (LID) to identify one of several versions of the same bundle, collection, or product. Use double colons to append the VID to the LID. The combination is called a *versioned identifier* (LIDVID). LIDVIDs are used to locate products within PDS; every version of every product within PDS has a unique LIDVID.

The following example LIDVIDs are based on the example LIDs in Section 5.2.1. In all cases the VID is 1.0.

5.4.1 Examples

Cruise Phase Bundle

Bundle LIDVID

```
urn:nasa:pds:ssc01.hirespc.cruise::1.0
```

Collection LIDVIDs

```
urn:nasa:pds:ssc01.hirespc.cruise:browse::1.0
urn:nasa:pds:ssc01.hirespc.cruise:context::1.0
urn:nasa:pds:ssc01.hirespc.cruise:data::1.0
urn:nasa:pds:ssc01.hirespc.cruise:document::1.0
urn:nasa:pds:ssc01.hirespc.cruise:xml_schema::1.0
```

Product LIDVIDs [data products for sclock = 31234567]

```
urn:nasa:pds:ssc01.hirespc.cruise:browse:browse_31234567::1.0
urn:nasa:pds:ssc01.hirespc.cruise:data:data_raw_31234567::1.0
urn:nasa:pds:ssc01.hirespc.cruise:data:data_derived_31234567::1.0
urn:nasa:pds:ssc01.hirespc.cruise:document:errata::1.0
urn:nasa:pds:ssc01.hirespc.cruise:xml_schema:table_character_0411f::1.0
```

When you have constructed draft LIDs and LIDVIDs contact your PDS node representative to verify that they are unique and conform to the rules.

6 Designing Labels

6.1 Overview of Labels and Schema

Labels are fundamental to PDS products; they describe both the content and format of products. They also allow links to be established among products, so that observational data can reference descriptions of the instrumentation that collected the bits, the spacecraft that hosted the instruments, and the organizations that supported the activity. They also help to bind related products into collections and related collections into bundles. Using labels written in XML, and constrained by an XML schema and a set of Schematron rules, helps ensure that PDS4 products are thoroughly and consistently documented, and that their metadata are available to the wide range of third-party software that reads and writes in XML.

The *PDS4 Information Model* [1], *PDS4 Data Dictionary* [3], and their XML representation in the *PDS4 Common Schema and Schematron* [4] serve as a library of generic definitions for each PDS4 product type. PDS also maintains data dictionaries for specialized disciplines such as geometry and cartography and for specific planetary missions; these are called Local Data Dictionaries or LDDs. They are also expressed as XML schemas for use in PDS labels. The PDS4 Schema and all current Local Data Dictionary schemas are available at https://pds.nasa.gov/pds4/schema/released/.

When a schema is used in a PDS4 label it is associated with a namespace. A namespace is a context for the terms defined in the schema. The common PDS4 schema has the namespace **pds**, and it is the default namespace in a PDS4 label. Other namespaces correspond to Local Data Dictionaries such as **cart** for the cartography dictionary, **disp** for the display dictionary, and **mvn** for the MAVEN mission dictionary. If classes or attributes from other namespaces are used in the label, the terms are prefixed with that namespace, as in **cart:map_projection_name** and **disp:vertical_display_direction**.

When you prepare data for delivery to PDS, you will certainly be involved in creating labels; you may also be involved in creating discipline- or mission-specific LDDs. Section 6 will focus on the components of labels, design choices you will need to make, and tools you can use. Basic information about XML and XML schemas can be found in Appendix D, specific instructions for editing PDS XML labels are in Appendix E, and a discussion about creating Local Data Dictionaries is in Appendix K.

6.2 Basic Product Labels

We can think of the material in the PDS archive as either observational or supplementary data (see Appendix A in *Concepts* [5] for formal definitions). We discuss labels for observational products first; then we discuss aspects of non-observational (supplementary) product labels, which differ slightly. Labels for aggregate products – that is, collections and bundles – are discussed in Appendix M.

You and your node representative should discuss the nature of the files that will be included in your archive, roughly categorizing the products according to whether they are observational or supplementary. The node representative can then identify the appropriate product types, such as Product_Observational and Product_Document, and their associated controlling files — common, discipline, and mission schema and Schematron files. This set of schemas and

Schematron files will be used to generate label templates and will provide the syntactic and semantic validation criteria for ensuring the integrity of the products in your archive. If you don't have them already, you can get them online at https://pds.nasa.gov/pds4/schema/released/ or the node representative can provide the files for you.

6.2.1 Product Type Selection

Each PDS4 label must identify the type, or class, of product the label is describing. The product type determines the set of required and optional classes and attributes to be specified in the product label. Product types, in alphabetical order, are as follows (*PDS4 Information Model* [1]):

- **Product_Ancillary** – a basic product containing data that are supplementary to observational data. Note that Product_Ancillary may be used only if no other supplementary product type is appropriate (See Appendix C).
- **Product_Browse** - a basic product containing a low resolution or "quick-look" version of an observational product.
- **Product_Bundle** - an aggregate product used to identify the member collections of an archive bundle.
- **Product_Collection** - an aggregate product used to identify the member basic products of an archive collection.
- **Product_Context** - a basic product identifying the physical (instrument, spacecraft, target, people) and conceptual (investigation, node) objects related to an observational product.
- **Product_Document** - a basic product identifying a single logical document, such as an interface specification, instrument description, or user's manual; the document product may be archived in multiple formats under the single logical Product_Document definition.
- **Product_File_Text** - a basic product consisting of a single digital file with ASCII character encoding.
- **Product_Native** - a basic product containing data in the original format returned by the observing system, used for data that cannot be described using one of the PDS4 formats for observational data. See *Standards Reference* [2] Section 9E for restrictions on the use of this product type.
- **Product_Observational** - a basic product comprising one or more images, tables, and/or other fundamental data structures that are the result of a science or engineering observation.
- **Product_SPICE_Kernel** - a basic product consisting of a SPICE kernel.
- **Product_Thumbnail** - a basic product consisting of a highly reduced version of an image, even smaller than a Product Browse image, typically used in displaying the results from search interfaces.

- **Product_XML_Schema** - a product consisting of XML formatted schemas, Schematron files, or any other reference schemas used in the interpretation of an observational product.
- **Product_Zipped** – a product containing other products and associated files, packaged using a PDS4-approved aggregation algorithm such as zip or tar.

The selection of the appropriate product class, when the choice is not obvious, is outlined in Appendix C, How To Select the Right Class for a Product.

6.2.2 Basic Product Labels for Observational Products

The science data products in your archive will be described using Product_Observational labels, so we'll go into more detail about them in this section.

6.2.2.1 Selecting the Structural Description for Observational Products

Based on the nature of the observational data, you and your node representative will determine which subclasses to use for describing the observational objects (e.g., **Array**, **Table**, **Header**, etc.) in your archive. The **Product_Observational** class allows the following subclasses (*PDS4 Information Model* [1]).

For binary array data:

- **Array**
- **Array_2D**
- **Array_2D_Image**
- **Array_2D_Map**
- **Array_2D_Spectrum**
- **Array_3D**
- **Array_3D_Image**
- **Array_3D_Spectrum**

For binary tabular data:

- **Table_Binary**

For character tabular data:

- **Table_Character** (fixed width fields)
- **Table_Delimited** (variable width fields)

The following are allowed, but are less frequently used, and special constraints may apply:

- **Header**
- **Encoded_Header**
- **Encoded_Binary** (Supplemental)

- **Encoded_Byte_Stream** (Supplemental)
- **Encoded_Image** (Supplemental)
- **Parsable_Byte_Stream** (Supplemental)
- **Stream_Text**

When a class with one or more levels of subclasses could be used to describe an object, the most specific subclass that is appropriate and allowed in the class must be used. For example, when a two-dimensional array contains image pixels, and **Array**, **Array_2D**, and **Array_2D_Image** are possible choices, **Array_2D_Image** must be used rather than **Array** or **Array_2D**.

6.2.2.2 Basic Product Label Organization – Observational Products

In a **Product_Observational** label there are several blocks of information called *areas*. Each area contains one or more classes, each of which may have several attributes. The areas, plus some XML overhead at the beginning, are shown in Figure 6-1 and explained in the text that follows. The figure omits the contents of each area for brevity; see Appendix F for the details.

- XML Prolog
  - The statement beginning with `<?xml` is an XML declaration. It means that the label is an XML versioned document.
  - The statement beginning with `<?xml-model` is a processing instruction. It identifies the Schematron file against which the label is validated. A label may have more than one of these statements.
- Root Tag
  - Provides a declaration of the root element of the label; the product type — in this case, **Product_Observational**.
  - Lines beginning with `xmlns:` identify the namespaces used in the label (i.e., the **pds** namespace plus any additional discipline or mission namespaces)
  - Lines beginning with `xsi:schemaLocation` match namespaces to the locations where the associated schemas are found.
- Identification Area
  - Provides identification information specific to the product (the LID and VID, the information model version, the product class, etc.)
  - Provides citation information specific to the product
  - Provides modification history specific to the product
- Observation Area
  - Provides information about the investigation, instrument, target, times, etc.
  - Includes subsections for relevant classes specific to one or more discipline dictionaries, and to the mission (or an equivalent namespace).

XML Prolog

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

Root Tag

```
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
   xmlns:pds="http://pds.nasa.gov/pds4/pds/v1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                       http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">
```

Areas

```
<Identification_Area>
      27 lines of detail
</Identification_Area>

<Observation_Area>
      76 lines of detail
</Observation_Area>

<Reference_List>
      10 lines of detail
</Reference_List>

<File_Area_Observational>
      41 lines of detail
</File_Area_Observational>
```

End Tag

```
</Product_Observational>
```

Figure 6-1. Example Label Structure for Product_Observational

- Reference List Area
  - Provides identification information for products, such as journal articles, that are relevant to understanding the product. References may be made to sources both internal and external to PDS.
- File Area
  - Identifies the file(s) containing the digital object(s), and
  - Includes classes that describe each digital object in the given file (e.g., the description and parameters of each header, table, image).
- End Tag
  - The end tag corresponds to and closes the Root Tag.

Any basic product may include multiple objects and multiple object types. For example, you may have a product comprising an **Array_2D_Image**, a **Table_Character** histogram of pixel values, and a **Header**. The three objects, which may or may not all reside in the same file, coupled with the single XML label that describes these objects, form the single digital product. If they are all in the same file, then the label's File area will have a **File_Area_Observational** class that permits inclusion of multiple digital objects (e.g., **Header**, **Table_Character**, **Array_2D_Image**). If the objects are in two or more files, then the label will have a separate **File_Area_Observational** class for each file. Ask your consulting node to recommend the best method for your products.

If your observational products have associated browse objects, you may treat them as separate browse products and associate them with the corresponding observational products using the **Reference_List** class in the Reference List area.

With two exceptions, all **Product_Observational** labels must have the structure shown in Figure 6-1 (*PDS4 Information Model* [1]). The exceptions are:

- The **Reference List** class is optional; and
- One or more **File_Area_Observational_Supplemental** classes, which allow supplementary data objects to be carried along with observational data products, may be added. An example of a label that uses the **File_Area_Observational_Supplemental** class is **thermal_neutron_map.xml** in the *DPH Examples* (https://pds.nasa.gov/pds4/doc/examples/).

This structural consistency simplifies extraction of information from such labels, facilitating search and retrieval of products by future users. Note, however, that there is considerable flexibility for content within the areas shown in Figure 6-1.

For additional information regarding the definitions and *cardinality* of the attributes and classes used within the definition of a basic product label, consult the *PDS4 Data Dictionary* [3].

6.2.3 Basic Product Labels for Non-Observational Products

Labels for non-observational products are structurally similar to those for observational products (Section 6.2.2); but there are more product types, and those product types have more variations. This section highlights the differences.

6.2.3.1 Selecting the Appropriate Types for Non-Observational Products

Common product types for non-observational products include:

- **Product_Ancillary**
- **Product_Browse**
- **Product_Context**
- **Product_Document**
- **Product_File_Text**
- **Product_Native**

- **Product_SPICE_Kernel**
- **Product_Thumbnail**
- **Product_XML_Schema**

These product types are defined in the *PDS4 Information Model* [1].

6.2.3.2 Basic Product Label Organization – Non-Observational Products

The label for a non-observational (that is, a supplementary) product is similar to the label for **Product_Observational**; but the many options for the different product types make a general description impossible. Supplementary product labels begin with the XML Prolog and Root Tag, and they end with the End Tag (see Figure 6-1). They include the **Identification_Area** and (optionally) the **Reference_List**; but, after that, the customization for each supplementary product type defies generalization. Start by selecting one of the non-observational product types listed above and look up that product class in the *PDS4 Information Model* [1] to see its components. You will need to work closely with your consulting node when designing these labels. Your node representative can provide examples of the product types to help you choose.

6.2.4 Aggregate Product Labels

There are two types of aggregate products:

- **Product_Collection**
- **Product_Bundle**

As mentioned in Section 4.1, a collection is a group of related products. A **Product_Collection** product is simply a list, or inventory, of the products in the collection. The products are listed by their LIDs or LIDVIDs. As it is possible for a product to belong to more than one collection, the inventory also tells whether this is the primary collection for a product or whether it is a product's secondary collection. A **Product_Collection** product is accompanied by an XML label.

Similarly, a bundle is a group of related collections. A **Product_Bundle** product is an inventory of the collections in the bundle. Unlike a **Product_Collection** product, a **Product_Bundle** product consists of the label only, not a label and a separate file. The inventory of collections is included in the label.

Instructions for creating **Product_Collection** and **Product_Bundle** labels are in Appendix M.

6.3 Data Dictionary Overview

6.3.1 What Is a Data Dictionary For?

A data dictionary has several purposes. First, the dictionary is a reference for users of the PDS, defining attributes and classes that describe planetary data. Second, the dictionary is a reference for data producers in designing data labels. Third, the dictionary ensures that attributes and classes in the data descriptions are used in a standard, consistent, and predictable manner.

Conceptually, a data dictionary defines the attributes and classes that may be used in PDS4 product labels. It is intended to be readable both by humans and by software. Practically speaking, it must contain definitions as well as the syntax and semantic constraints placed on values of attributes. For classes, it provides the explicit list of attributes defining the class and indicates which are required, optional, and repeatable.

Every attribute and class that is used in any PDS label must first be defined in a data dictionary.

Data dictionaries are classified as:

- Common
- Discipline-specific
- Mission-specific

The common dictionary is the fundamental PDS4 Data Dictionary, which is represented using the **pds** namespace in labels, schemas and Schematron files. Discipline-specific and mission-specific dictionaries are known as Local Data Dictionaries (LDDs). All PDS archives are based on the common dictionary; they may also include classes and attributes defined in one or more PDS discipline and/or mission dictionaries.

The list is hierarchical. Discipline and mission dictionaries reference the common dictionary and any other relevant discipline dictionaries. However, a mission-specific dictionary is not referenced by any other dictionary.

Discipline-specific data dictionaries are produced by PDS. These include dictionaries intended to support archives relevant to specific discipline nodes (e.g., an Atmospheres Node dictionary managed by the Atmospheres Node) and dictionaries that support cross-discipline concepts such as geometry and cartography.

Mission-specific data dictionaries are produced by PDS in consultation with mission personnel who are responsible for archive development. They contain attributes and classes specific to a particular mission or investigation. For example, the MAVEN mission data dictionary includes the attributes **mission_phase_name**, **mission_event_time_start_count**, and **mission_event_time_stop_count**, among others. A mission may have a single dictionary or multiple dictionaries; for example, it may have one dictionary for instrumentation descriptors and another for data descriptors. In general, mission personnel may manage mission dictionaries, but consulting nodes participate actively in design and prototyping to ensure conformance with PDS standards.

6.3.2 Why You Need to Understand Data Dictionaries

During the process of designing labels for your observational and non-observational products, you may discover that you want to include certain items of information in the labels but you can't find any relevant attributes or classes for them in the PDS4 common data dictionary. The first thing to do is ask your PDS node representative, in case you have overlooked the classes or attributes you need. The next step is to look for the relevant definitions in one of the existing Local Data Dictionaries. For example, if you ae labeling an image and you want to include map projection information, the Cartography LDD is the logical place to look. Or, if you want to include mission-specific information in the label, look for a Local Data Dictionary for the mission. You'll find the currently available Local Data Dictionaries online at

https://pds.nasa.gov/pds4/schema/released/. Each LDD is given in three parts: an XML schema file (extension .xsd), a Schematron file containing validation rules (extension .sch), and a PDS4 label that describes them both (extension .xml).

It's possible that the new attributes or classes you want rightly belong in one of the existing LDDs. In this case you and your PDS node representative should contact the PDS node that maintains the LDD to see whether the desired attributes and/or classes can be added.

It's also possible that the attributes or classes you want do not belong in either the common PDS4 Data Dictionary or any of the current LDDs. In this case you'll need to create your own LDD to define these items. This is a non-trivial task that will require help from your PDS node representative.

Appendix K has detailed instructions for creating and using LDDs.

6.4 Label Design Tools

PDS provides and recommends tools to help you design labels, validate labels, produce many labels at one time, and create a Local Data Dictionary. You may use these tools or any others that you prefer. This section discusses label design tools. Validation tools are discussed in Section 11.5, bulk label generation tools in Section 7, and Local Data Dictionary tools in Appendix K. PDS tools are available on the PDS4 Software web page at https://pds.nasa.gov/pds4/software/. User-contributed tools are available in the PDS Tool Registry at https://pds.nasa.gov/tools/tool-registry/.

6.4.1 XML-Aware Editors

When you are ready to begin designing XML labels for your data products, you do not need to start with a blank slate. One of the advantages of having the PDS4 Information Model expressed as an XML schema is that an XML-aware editor can read the schema and automatically generate a draft label for you. You can then add and subtract parts of the label as needed and fill in some of the blanks. The editor will alert you if you make a change that is not supported by the model.

You can create a complete, valid PDS4 label in this way. You can also create a label *template* – that is, a label with some placeholders that can be filled in later by software. For instance, you could draft a simple label with placeholders for the product LID, VID, and file name. You would then use software (see Section 7) to create a label for each data product by starting with the template and filling in the missing information for each product.

Two XML-aware editors are commonly used within PDS, the oXygen editor (http://www.oxygenxml.com) and the Eclipse editor (http://www.eclipse.org). Both of these make it easier to view XML files, which can appear daunting on your first encounter. They offer color highlighting and different graphic views of an XML file, as well as instant validation of the label based on its associated schema. Eclipse is an open source product available for free. It is the framework that underlies several commercial products, including oXygen. As you might expect, oXygen has more functionality and is generally easier for new users to learn than the free Eclipse editor.

It's also possible to use a standard text editor to read and write XML labels. Some provide XML highlighting and other options. Most popular web browsers can display XML files with some

kind of helpful formatting. If yours doesn't immediately recognize and format the XML, try using the View Source option in the browser.

Your initial draft label will be based on the type of product to be described. For example, if you are labeling a two-dimensional image, you'll start with a generic **Product_Observational** label with the subclass **Array_2D_Image**. Your PDS node representative can create the initial draft label for you, or you can do it yourself following the instructions in Appendix E. Once you have the initial draft, you may modify it to suit your needs. Details about editing a label are in Appendix E.

6.4.2 Copying a Similar Label

Another way to draft a label is to start with a completed PDS4 label for a similar product. Your PDS representative can help you find one. It's best to use a recent label that relies on the current PDS Information Model, and one that uses the product class that you need. You will still find an XML-aware editor to be very useful for modifying the existing label to fit your needs. Details about editing a label are in Appendix E.

7 Mass Producing Labels

If you are developing an archive with more than just a few products, you will want to automate label generation as much as possible. A common approach is to first generate a label template and then use a software tool to fill in the template to create a label for each data product. A label template looks like the final label except that it contains placeholders that software will replace.

7.1 Creating a Label Template

A good way to create a label template is to start with a valid label for a particular data product, and then generalize by inserting placeholders for those values that will change from one product to the next. Appendix E and Appendix F walk you through the process of creating a label using an XML-aware editor and then modifying it to create a label template.

7.2 The PDS Generate Tool

PDS has a Generate Tool that generates PDS4 labels from either a PDS3 label or a PDS3-specific Document Object Model[9] (DOM) object. This is useful if you are migrating PDS3 data to PDS4 rather than building a new PDS4 archive from scratch. The Generate Tool allows you to generate PDS4 labels having values that map directly to the keywords and keyword values in the original PDS3 label. For more information, go to:

https://pds.nasa.gov/pds4/software/generate/

7.3 PDS Node-Specific Tools

Several of the PDS Nodes have created their own tools for generating labels from a label template. Ask your consulting node for a recommendation.

7.4 The PDS Tool Registry

The PDS Tool Registry at https://pds.nasa.gov/tools/tool-registry/ accepts tools submitted by PDS users. Submissions to the Tool Registry are vetted by the PDS Engineering Node. Check the registry for label-generation tools.

Commented [Office2]: There are too many tools to list, but a selection with one-line descriptions of tools that could be valuable in archive generation, assembly, and valiudation could be helpful.

Commented [SS3R2]: Actually none of the tools in the registry does label generation except the Generate tool already mentioned. Maybe "IGPP Docgen" from PPI, although it's not obvious from the description.

[9] The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of the document and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content.

8 Documenting the Archive

Archives include documentation to augment the information furnished in the product labels and provide assistance in understanding the data products and accompanying materials. Typical archive documents include:

- Relevant flight project documents
- Instrument papers
- Science articles
- Software Interface Specifications (SISs)
- Software user manuals

Criteria for inclusion of a document in a PDS archive are:

- The document is necessary (or at least useful) for evaluating, understanding, and using the data.
- Document distribution is not restricted.

A scientist who is familiar with the field, but not necessarily with the observing system or data, should be able to understand and use the data based on information contained in labels, in documents within the archive, and in external documents referenced by the archive.

8.1 Documents Included in the Archive

Your archive bundle will probably include a document collection. Each document in the collection will be described by a PDS4 label. When designing these labels for your documents, you may use the **Product_Document** and/or **Product_File_Text** classes (see Section 6.2.1). Documents must be in a PDS-compliant format — UTF-8 text, PDF/A-1a (which is preferred), or PDF/A-1b (see Section 4.3.3). Document labels are fairly simple to create compared to labels for observational products, but you can always get help from your consulting node representative. Figure 8-1 shows how to use the **Product_Document** class to describe a document. You may see the full label in the *DPH Examples* [7] online at https://pds.nasa.gov/pds4/doc/examples.

Documents prepared for inclusion in an archive must conform to the *Policy on Formats for PDS4 Data and Documentation* at https://pds.nasa.gov/policy/format_policies_final.pdf

Documents in an archive are expected to meet not only the PDS label and format requirements, but also the structural, grammatical and lexical requirements of a refereed journal submission. Documents submitted for archiving that contain spelling errors, poor grammar or illogical organization may be rejected and may ultimately lead to the rejection of the submitted data for lack of adequate documentation.

```
<Product_Document xmlns="http://pds.nasa.gov/pds4/pds/v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                     http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">

    <Identification_Area>
        [17 lines omitted]
    </Identification_Area>

    <Document>
        <publication_date>2016-06-16</publication_date>
        <description>Description of the of the MER Pancam
                     photometry cube dataset.</description>
        <Document_Edition>
            <edition_name>MER Pancam Photometry Dataset Description
            </edition_name>
            <language>English</language>
            <files>1</files>
            <Document_File>
                <file_name>pancam_photometry_archive_description.pdf</file_name>
                <document_standard_id>PDF/A</document_standard_id>
            </Document_File>
        </Document_Edition>
    </Document>
</Product_Document>
```

Figure 8-1. Example Product_Document Label

8.2 Documents External to PDS

To ensure the integrity of the archive, the preferred approach is to have all documentation within the archive, but in some cases this is either impractical or impossible. For example, when a copyright holder will not allow a document to appear in the archive, a reference to the document must be supplied instead. The reference should appear in the label of the product to which the document applies.

Use the **External_Reference** class to cite a document that resides outside of PDS. This class occurs inside the **Observing_System_Component** class in your **Product_Observational** label. It also occurs inside the **Reference_List** class, which can be used in many types of labels. You can include any number of **External_References** and **Internal_References** in a **Reference_List**.

The **External_Reference** class provides a complete bibliographic citation to a published work, optionally including its Digital Object Identifier (DOI) and a description of the work. Figure 8-2 shows an example of an **External_Reference** to a published paper.

```
<Reference_List>

    <Internal_Reference>
        <lidvid_reference>
            urn:nasa:pds:joe_investigator_spectra:calibration:calib_foil::1.0
        </lidvid_reference>
        <reference_type>data_to_calibration_product</reference_type>
        <comment>Spectrum of 25 micrometer Fe foil
                 used for calibration</comment>
    </Internal_Reference>

    <External_Reference>
        <doi>doi:10.1002/2016JE987654321</doi>
        <reference_text>Investigator, J.O., 2017, Spectral Properties
            of Laboratory-Synthesized Glasses, Journal of Geophysical
            Research - Planets, doi:10.1002/2016JE987654321</reference_text>
    </External_Reference>

</Reference_List>
```

*Figure 8-2. Examples of **Internal_Reference** and **External_Reference***

8.3 Documents Elsewhere in PDS

Just as the **External_Reference** class is used to cite a reference outside of PDS, the **Internal_Reference** class may be used to link to a document inside PDS, using the document's LID or LIDVID. Figure 8-2 also shows an **Internal_Reference** to a calibration product using the LIDVID for the product.

Notice that an internal reference must include both a LID (or LIDVID) and a reference type. You can look up the appropriate value for reference type in the online *PDS Information Model* [1] (https://pds.nasa.gov/pds4/doc/im/current/) by searching for "reference type in Internal_Reference".

8.4 Restricted Documents

Certain documents have restricted distributions and therefore cannot be included in PDS archives, which are accessible to anyone with an Internet connection. The most common situations involve documents protected by copyright (when the copyright holder has not given permission for re-use) and documents covered by United States International Traffic in Arms Regulations (ITAR). Both circumstances are covered by United States law, and violations could lead to serious consequences for both PDS and the individuals involved. Discuss the issues with your consulting node should it appear that your documents may be restricted; your consulting node may wish to raise the question to higher levels within PDS and NASA.

Copyright issues are the easiest to resolve. If the documents in question are available to the public through the copyright holder, then **External_Reference** is the appropriate way to cite the document. The data user can negotiate access to the documents with the copyright holder, possibly in exchange for a fee.

ITAR covers a wide range of hardware, software, and operations topics in the weapons, missiles, and technology areas. In general, the results of science investigations are not subject to ITAR control, but the performance, design, and operation of the equipment may be. Documentation concerning the construction and operational details of new state-of-the-art instrumentation are likely candidates for review. NASA centers (e.g. JPL and GSFC) and institutions heavily involved in technology (e.g. JHU/APL) often have ITAR review teams, which can determine whether questionable documents are suitable for release and certify them as such. If not suitable, investigation teams may have to rewrite the documents; determining when redaction has a negative impact on the ability of a science user to understand and work with the data can be challenging.

In the end it is the responsibility of the data provider, not PDS, to ensure that archive documents submitted to PDS are unrestricted.

9 Creating and Using Context Products

9.1 What Are Context Products?

As mentioned before in Section 2.6, context products provide a way to associate the material in an archive with relevant information outside the archive – to place an archive in the context of the whole of PDS.

There are context products for missions, spacecraft, instruments, laboratories, observatories, telescopes, planetary bodies, PDS nodes, and a few other things. A context product is an exception to the general rule that a product consists of a digital object and a PDS label; a context product consists of a label only. It contains a unique identifier for the physical or conceptual object (i.e. the mission, spacecraft, etc.), a brief description of the object, and an optional list of references that provide more information about the object.

All context products belong to the master context bundle, which is maintained by the PDS Engineering Node. If you are submitting an archive to PDS that requires new context products – that is, your archive is related to a mission, spacecraft, instrument, etc., that is not already in the master context bundle – you will need to work with your consulting node representative to create the new context products.

Figure 9-1 is the context product for the target (planet) Mars.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1301.sch"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
<Product_Context
    xmlns="http://pds.nasa.gov/pds4/pds/v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                        http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1301.xsd">

    <Identification_Area>
        <logical_identifier>
             urn:nasa:pds:context:target:planet.mars
        </logical_identifier>
        <version_id>1.0</version_id>
        <title>MARS</title>
        <information_model_version>1.3.0.1</information_model_version>
        <product_class>Product_Context</product_class>
        <Modification_History>
            <Modification_Detail>
                <modification_date>2014-11-21</modification_date>
                <version_id>1.0</version_id>
                <description>
                    extracted metadata from PDS3 catalog and
                    modified to comply with PDS4 Information Model
                </description>
            </Modification_Detail>
        </Modification_History>
    </Identification_Area>

    <Target>
        <name>MARS</name>
        <type>Planet</type>
        <description>
          [lines omitted]
        </description>
    </Target>
</Product_Context>
```

Figure 9-1. Example of a Context Product: Target Mars

9.2 How Context Products are Generated

When creating a new context product it is easiest to modify an existing one of the same type. Look in the PDS4 master context bundle, which is available at https://starbase.jpl.nasa.gov/pds4/context-pds4/. Alternatively, use an XML-aware editor to create a draft context product label from the PDS4 common schema, using the method described in Appendix E. When you have your draft label, use the instructions in Appendix F to modify it.

The most important element in a context product is its LID. If you need to create a new context product, you should review the guidelines given in Appendix L for forming a context product

LID. You should also look at the list of currently approved context product LIDs at [Ron and/or Richard to provide link]. You or your consulting PDS node will need to submit your proposed LIDs to a data engineer at the PDS Engineering Node to make sure they are unique and acceptable.

9.3 How Context Products are Maintained

Context products are maintained in the master context bundle by a data engineer at the PDS Engineering Node. It is not necessary to include them in your archive bundle, although you may choose to do so. If you do, then list them as secondary members of a context collection in your bundle. See *Standards Reference* [2] Section 8B.3 for details.

If a change to a context product is needed, notify the data engineer responsible for maintaining the master context bundle. Ask your consulting node representative if you don't know who the data engineer is.

9.4 How Context Products are Used

The **Observation_Area** section of a **Product_Observational** label includes several places where an **Internal_Reference** is called for. In each case the LID in the **Internal_Reference** is the LID of a context product. This is how a data product is connected to a mission, spacecraft, instrument, and target. Figure 9-2 is a label excerpt from the *DPH Examples* [7] online showing the **Observation_Area** of a **Product_Observational** label.

You will also find **Internal_References** to context product LIDs in the label for a collection product. This is how a collection is associated with a mission, spacecraft, instruments, and targets.

You may run into other kinds of labels with an **Internal_Reference** class that requires a context product LID, but these are the most common uses.

```
<Product_Observational [9 lines omitted]>

  <Identification_Area> [6 lines omitted] </Identification_Area>

  <Observation_Area>
    <Time_Coordinates>
      <start_date_time>2005-12-30T18:06:12.650Z</start_date_time>
      <stop_date_time>2005-12-30T18:07:35.467Z</stop_date_time>
    </Time_Coordinates>
    <Investigation_Area>
      <name>Mars Exploration Rover</name>
      <type>Mission</type>
      <Internal_Reference>
        <lid_reference>
          urn:nasa:pds:context:investigation:mission.mars_exploration_rover
        </lid_reference>
        <reference_type>data_to_investigation</reference_type>
      </Internal_Reference>
    </Investigation_Area>
    <Observing_System>
      <name>MER 2 Pancam</name>
      <Observing_System_Component>
        <name>MER2</name>
        <type>Spacecraft</type>
        <Internal_Reference>
          <lid_reference>
            urn:nasa:pds:context:instrument_host:spacecraft.mer2</lid_reference>
          <reference_type>is_instrument_host</reference_type>
        </Internal_Reference>
      </Observing_System_Component>
      <Observing_System_Component>
        <name>Pancam</name>
       <type>Instrument</type>
       <Internal_Reference>
         <lid_reference>urn:nasa:pds:context:instrument:pancam.mer2</lid_reference>
         <reference_type>is_instrument</reference_type>
       </Internal_Reference>
      </Observing_System_Component>
    </Observing_System>
    <Target_Identification>
      <name>Mars</name>
      <type>Planet</type>
      <Internal_Reference>
        <lid_reference>urn:nasa:pds:context:target:planet.mars</lid_reference>
        <reference_type>data_to_target</reference_type>
      </Internal_Reference>
    </Target_Identification>

    <Discipline_Area> [67 lines omitted] </Discipline_Area>
  </Observation_Area>

  <File_Area_Observational> [487 lines omitted] </File_Area_Observational>
</Product_Observational>
```

*Figure 9-2. Example of Context Product LIDs in **Product_Observational** Label*

10 Assembling the Whole Archive

Your consulting PDS node will probably require you to submit a fully assembled archive when the time comes for a peer review. Even if you have been validating the individual labels as you went along, it's best to validate the whole archive before submitting it to peer review. We'll discuss validation in Section 11. This section explains what a fully assembled archive looks like.

Recall that the design of your archive (Section 4.3) defined the bundle and collections it would contain, the directory organization, and the directory and file naming schemes. Start assembling your archive by building a directory tree to hold the pieces. In particular, if you have more than about 20 data products, consider how to divide your many data files and their labels into subdirectories. The directory tree usually mimics the bundle and collection hierarchy, but it doesn't have to.

Figure 10-1 is a diagram of a typical directory tree. This example uses directory names in uppercase and file names in lowercase just for illustration; this is not required. You can also look at the example archive online at https://pds.nasa.gov/pds4/doc/examples/.

Section 2.B of the *SR* [2] describes a suggested organization for assembling a PDS4 bundle for transfer to the PDS node that will permanently house your bundle. This transfer organization may be temporary, in that the archiving node may want to rearrange some aspects of your bundle in order to integrate it into their overall holdings. At the top level is the bundle product. It's a label for the bundle as a whole. It may have any name starting with **bundle**, such as **bundle_raw.xml**, but plain **bundle.xml** is fine. A readme file, if present, can be an overview of the contents of the bundle. It's described in **bundle.xml**.

There are no requirements for specific collections in a bundle, not even a data collection (you could have a bundle containing only documents, for example.) But if the bundle does contain a data collection, it is required to be in a directory whose name starts with **data** (*SR* [2] Section 2B.2.2). Similarly, if the bundle contains browse or document collections, they must be in directories whose names start with **browse** or **document**. See Section 2 of the *Standards Reference* for other prescribed directory names. Subdirectories beneath these directories can be named as you see fit. For the data directory, if you have more than about 20 data products, they should be divided among subdirectories. If you have browse products you'll probably find it convenient to store them in a structure that matches the data directory structure.

For each collection there must be a collection inventory table and its label, and these must be in the top level of the collection's directory. Their names must start with **collection**. A collection inventory is simply a table listing the LIDs or LIDVIDs of the products in the collection. As it is possible for a product to belong to more than one collection, the collection inventory also tells whether each product is a primary or secondary member of the collection. (It's a secondary member if it is already a primary member of another collection.) The collection inventory label describes the inventory table, of course, but it contains additional information about the collection that must follow the specific requirements in the *Standards Reference* [2] Section 9C. Instructions for generating collection inventories and labels, along with bundle labels, are found in Appendix M.[10]

[10] If your collection meets the criteria for a Mission Science Data Collection defined in the *Standards Reference* [2] Section 9C.3.1, then your collection inventory label must include the metadata listed in that section.

```
readme.txt
bundle.xml
DATA
        collection_data_inventory.csv
        collection_data_inventory.xml
        YEAR1
                data_2017_01_01.csv
                data_2017_01_01.xml
                …
        YEAR2
                data_2018_01_01.csv
                data_2018_01_01.xml
                …
BROWSE
        collection_browse_inventory.csv
        collection_browse_inventory.xml
        YEAR1
                browse_2017_01_01.jpg
                browse_2017_01_01.xml
                …
        YEAR2
                browse_2018_01_01.jpg
                browse_2018_01_01.xml
                …
DOCUMENT
        collection_document_inventory.csv
        collection_document_inventory.xml
        errata.txt
        errata.xml
        userguide.pdf
        userguide.xml
```

Figure 10-1. Example archive directory structure

It's worth repeating that a context collection is not required. If your bundle has not introduced any new context products to PDS4 (see Section 9), there is no reason to include a context collection. Even if your bundle does require new context products, they may be delivered separately to the PDS Engineering Node, as negotiated with your consulting node. See the *Standards Reference* Section 8B.3. If you do include a context product collection in your bundle, the products should be listed in the context collection inventory table as secondary products. They are primary products in the PDS4 master context bundle.

When you have built your bundle's directory structure, you may populate it with the files that belong in each directory, and update the collection inventory tables accordingly. When all the pieces are in place you can then validate the bundle. This is discussed in Section 11, Archive Validation.

11 Archive Validation

Archives submitted to PDS must be shown to be valid before they can be accepted. Validation means ensuring that the archive is complete, internally consistent, and consistent with other PDS archives and that the products in the archive are documented, correctly labeled, and error-free.

The PDS node that receives your delivery will perform most of these validation checks (except for science validation, as explained in Section 11.3). If errors are found, you will have to correct them and re-deliver. It's best to run the PDS-supplied validation tools on your bundles before delivering them in order to avoid last-minute corrections.

11.1 Label Validation

To be compliant with PDS4 standards an XML label must:

- Be both semantically and syntactically correct.
- Be compliant both in content and in referential integrity.
- Be compliant with the class and attribute structures defined by the PDS4 common schema and any relevant discipline and mission dictionary schemas.
- Be compliant with the rules governing specific attributes and their values as set by the PDS Schematron files and any relevant discipline and mission Schematron files. (See Appendix J for more about validation using Schematron files.)
- Properly describe the structure of the digital objects.

11.2 Data Product Validation

Validation against the schemas and Schematron files demonstrates only that a label is correctly formed and populated; it does not show that the label correctly describes its data product. You need to ensure that the data product can be read or displayed correctly using only the information in the label. For large deliveries it may be impractical to apply this test to every product, but you should at least do spot checks of each type of product. This kind of validation needs to be done by the data provider, who knows what to expect in a data product and would be able to recognize an error caused by misreading the file. The receiving PDS node will also do spot checks of the data.

You may test your labels and data products using PDS tools (see below) or using your own label-reading software that you would normally use to work with the data. If your archive contains browse data, they may be useful for comparing to the observational data as a test to see if the latter are being read correctly.

11.3 Science Validation

Science validation means checking that the data products in your archive contain the data you expect them to contain. This can be done by spot checking products, as just mentioned, but the best way to validate the science content of your data is to use the products. That is, the products that go into the archive should be *the same products that you use for research*, so that you are likely to uncover any anomalies in them in the course of your science analysis. This kind of validation can be performed only by the scientist providing the data. PDS personnel are not

likely to put the data products to use in a research environment, and even then they might not be able to recognize a problem with the data.

11.4 Archive Validation

The archive as a whole must be complete, internally consistent, and consistent with other PDS archives. To be complete, it must have all the products, collections, and bundles that it should have according to both its documentation and also the peer review results. Of course, a mission-based archive that accumulates over many deliveries may not be complete until the end of the mission, but the products, collections, and bundles expected for a particular delivery must be complete.

To be internally consistent, all products must belong to at least one collection and must be listed in the collection's inventory. All collections must belong to at least one bundle and must be listed in the bundle label. LIDVIDs must be correctly formed and unique.

To be consistent with other PDS archives, internal references must lead to documents and context products that actually exist.

11.5 Validation Tools

11.5.1 XML-Aware Editors

The best way to ensure a valid label is to use an XML-aware editor (see Section 6.4.1). These editors can show you at a glance where the label is not compliant with the schema(s). This is useful as a first step in validation, but keep in mind that XML-aware editors validate only the contents of the label, not the data file.

11.5.2 The PDS4 Validate Tool

The PDS4 Validate Tool is used for validating PDS4 product labels, collections and bundles. It can be found here:

https://pds.nasa.gov/pds4/software/validate

The Validate Tool performs the following checks:

- Checks whether a label complies with the schemas and Schematrons specified in its XML prolog and root tag
- Checks that files referenced in the label exist
- Checks that file names in the label have the same case as the actual file names
- Checks that file checksums in the label, if any, match the actual file checksums
- Checks that file checksums in the Checksum file, if any, match the actual file checksums
- Checks the referential integrity of LIDs and LIDVIDs, including
    - Is the LID valid?
    - Is the LID listed in the collection inventory product?

Commented [SS4]: This is based on what Ron said that Sean said, mostly, but Sean should review, because some of this may be wishful thinking on my part. It's not documented anywhere that I can find.

  - Are all LIDs listed in the collection inventory present in the collection?
  - For a collection product LID: is it listed in the bundle label?
  - Are all collection product LIDs listed in the bundle label present in the bundle?
  - For a context product LID: does a context product with this LID exist?
- Checks that any Local_Internal_Reference classes in the label refer to local identifiers defined elsewhere in the label

Currently the Validate Tool does not check that the label correctly describes the contents of the data file.

11.5.3 The Small Bodies Node PDS4 Viewer and Other Tools

The Small Bodies Node maintains a set of tools for reading and manipulating PDS4-labeled data at http://sbn.astro.umd.edu/tools/tools_readPDS.shtml. While most of the tools are intended for use by programmers, the set also includes the PDS4 Viewer, a standalone program that reads and displays PDS4 images, tables, spectra and arrays, at http://sbndev.astro.umd.edu/wiki/PDS4_Viewer. This tool can be useful for verifying that a label correctly describes its data object.

11.5.4 The PDS Transform Tool

The PDS Transform Tool converts PDS3 and PDS4 labels and data into other formats, including JPEG, GIF, PNG, TIFF, and others.

https://pds.nasa.gov/pds4/software/transform/

11.5.5 Other Tools Maintained by PDS

Other label-reading tools maintained by the PDS Engineering Node include software libraries, which can be found here:

https://pds.nasa.gov/pds4/software/

11.5.6 Tools Contributed by PDS Users

PDS maintains a Tool Registry at https://pds.nasa.gov/tools/tool-registry/ to which PDS users may submit tools they have developed and download tools submitted by other users. Submissions to the Tool Registry are vetted by the PDS Engineering Node.

Commented [Office5]: There are too many tools to list, but a selection with one-line descriptions of tools that could be valuable in archive generation, assembly, and validation could be helpful.

Commented [SS6R5]: The only PDS4 tools in the registry that would be useful for archive development and that have not already been mentioned are the Data Dictionary Search and possibly IGPP Docgen.

12 Peer Review

12.1 Why Peer Review Is Needed

Peer review is the key to the integrity of the PDS holdings. Every archive submitted to PDS must pass a peer review before it can be accepted. Archive peer review is analogous to the review of an article submitted to a scientific journal, although the procedure is different. Your consulting PDS node representative will coordinate the peer review of your bundle by soliciting reviewers, setting a schedule, making the bundle available, collecting review comments, and coordinating your response to the comments.

The review committee will include scientists who are experienced in the types of data in the bundle and, preferably, have a particular interest in your data. In the case of a mission-related bundle, the ideal mix of science reviewers includes some from outside the mission, some from within the mission but outside the instrument team, and possibly some who are involved in the team but not in the data product design. This last group may be the most critical of reviewers, having a real stake in the outcome. The committee will also include at least one PDS reviewer from another node and a PDS data engineer from the Engineering Node. Finally, the data provider(s) are included in the review committee. Reviews are held openly; data providers know who the reviewers are, all comments are shared with the whole committee, and members are expected to discuss any serious issues in order to reach a resolution.

When a reviewer submits a comment that recommends a change to the review material, or asks a question about it, the comment is called a *lien*. The review coordinator keeps a master list of all liens raised during a review. The data provider is asked to respond to each lien by stating what action, if any, will be taken to resolve it. The data provider is not required to do everything the reviewers recommend; but, if a recommendation will not be followed, the data provider must explain why and the reviewer must accept that explanation. The period of time during which the data provider is working on the review recommendations is called *lien resolution*.

As a rule the archive is accepted by PDS when all liens have been resolved. If outstanding liens do not materially affect scientific use of the data, the archive may be declared *certified* before all liens have been resolved. Once certified, the archive may be released online. Archives distributed from PDS web sites should be marked as Certified Data, or the web site should make the statement that all available archives are Certified Data unless otherwise indicated. This classification is significant because only Certified Data in PDS are eligible for use in research funded by some NASA programs. See the PDS Policy Defining Certified Data on the *PDS Policies* web page, https://pds.nasa.gov/policy/. Note that delivery is not complete until all liens have been resolved regardless of whether the data have been certified.

12.2 Typical Review Procedure

The approach to conducting a peer review varies depending on whether the archive accumulates over a long period of time, as for a planetary mission, or is completed in a single delivery, as for observatory or laboratory data.

Usually individual data providers are funded to generate an archive and deliver it to PDS in a single delivery near the close of the funding period. In this case the complete archive should be submitted to peer review sufficiently in advance of the end of funding to allow changes to be

made if needed. Your PDS representative will work out a schedule with you. With a single delivery, the products that are reviewed are the products that will be archived.

In the case of a mission that will deliver many times over several years, it is not practical to conduct a review for every delivery, nor to wait until the end of the mission to review the complete archive. The review must take place in time for any recommended changes to be made to the processing pipeline software, well before routine data handling begins. Therefore this kind of review covers a representative sample of pipeline data products and their documentation — typically a Data Product Software Interface Specification (SIS) document[11]. When actual data are delivered, they are validated by the receiving node for compliance with PDS standards and for compliance with the peer-reviewed documentation.

In general, the procedure is as follows, although it may vary slightly among PDS nodes.

1. The data provider delivers the archive (or a representative sample) to the coordinating node. Node personnel confirm that the delivery is suitable for review.
2. The review coordinator assembles a list of potential reviewers and invites them by email.
3. The review coordinator announces the opening of the review and provides access to a password-protected web site where the review materials are posted.
4. Reviewers have about two weeks to examine the materials and email their comments to the review coordinator.
5. The review coordinator posts comments (either separately, or collated in one list, or both) on the web site.
6. The data providers have about two weeks to email their response to the comments, which the review coordinator posts on the web site.
7. The review coordinator decides whether there is reason to hold a review teleconference, based on the data providers' response, for instance if there is an issue that cannot be resolved by email. If so, a teleconference is arranged at a time convenient for reviewers, data providers, and PDS representatives.
8. The review coordinator posts a list of the review liens (comments that require an action). At the teleconference, if there is one, the unresolved liens are discussed one by one and decisions are made for their resolution; otherwise the data providers simply begin working on the liens. Data providers may decline to follow a reviewer's recommendation if they can justify doing so (for example, because of insufficient resources), as long as the reviewer agrees and the resulting archive would still be PDS-compliant. A record of the resolution of each lien is posted on the review web site and retained by the PDS node.
9. The data provider resolves the liens within a given period of time, say from two weeks to a month. (The schedule should take into consideration other mission activities going on; in particular the liens should be resolved before delivery testing begins.)

[11] A SIS document is usually a mission deliverable. PDS does not require an archive to include a SIS; the requirement is only for adequate documentation to use and understand the archive. However, if an instrument team has to write a SIS anyway, they may include it in the archive if it helps meet the PDS documentation requirement (*Standards Reference* [2] Section 8A).

10. The data provider delivers revised materials to the review coordinator, who posts them on the review web site.
11. The review coordinator invites the reviewers to examine the revised material to be sure their comments have been addressed. If the reviewers are satisfied, the review is complete.

A representative archive that is to be delivered as real data in many installments – e.g., a mission archive – will not be made public because it is not an actual delivery; it contains only examples of data products. Actual deliveries will take place according to a schedule agreed upon between the mission and PDS. See Section 13 for details of how these archives are to be delivered to PDS.

13 Archive Delivery

The organizational products of PDS4 (i.e., bundles and collections) are logical, not physical, structures. However, products transferred to, within, or from PDS need to be placed into a physical structure. PDS uses delivery packages to accomplish such transfers.

In most cases it is convenient to organize delivery packages into a physical structure that parallels the logical structure of the archive (i.e., the bundle product at the root level with subdirectories for collections, etc.). However, alternate organizations are possible (such as flat directory structures for incremental deliveries of accumulating collections). PDS only requires that the sender and receiver agree on the structure in advance and that an MD5 checksum be provided for each file transferred (*PDS Policy on Use of Checksums*, PDS Policies page, https://pds.nasa.gov/policy/).

This section describes the typical procedures involved in delivering data to PDS. In practice, you should work with the discipline node that will receive the delivery to agree on the procedures that you will use.

13.1 Contents of the Delivery to PDS

A delivery to PDS may have up to three components:

1. The delivery package containing the material being transferred,
2. The checksum file, a list of individual MD5 checksums for every file in the delivery package.
3. The optional delivery manifest, which maps each product's LIDVID to the name of the product's label file, and

13.1.1 The Delivery Package

The term *package* is used to indicate the batch of archival material being transferred. A delivery package may come as a compressed file, e.g. with Zip, gzip or tar. It may come on an external drive delivered through the mail, or by electronic transmission such as FTP. The data provider and receiver determine the best packaging options for the delivery. In the absence of a transfer agreement, PDS provides a default procedure (see *PDS Standards Reference* [2], Section 2B.2) which is often recommended by discipline nodes.

13.1.2 The Checksum File

The checksum file is a file provided with each transfer to, from, or within PDS. The purpose of the checksum file is to allow the recipient to verify that the files have been delivered intact. By computing checksums on the files after delivery and comparing them to the original checksum file, the recipient will know if any errors have been introduced. The checksum file is required for every delivery.

The checksum file lists a checksum for *every* file in the package (not just every label file). If the delivery package is a zip file, tar file, etc., the checksum file must contain an entry for every file in the package after the package has been unpacked. Currently PDS uses MD5 checksums, so the checksum manifest is output from an MD5 checksum utility such as md5deep. Each entry

consists of a 32-character hexadecimal MD5 checksum value, followed by two spaces, followed by the file specification name relative to the root directory of the package, followed by an ASCII carriage-return line-feed pair.

The example below shows the structure of a checksum file:

```
6dab19e22439a419e3e9b9f4046bf390  ./browse/collection_browse.xml
bcb852a6a9292e304a93668e6cc2068c  ./browse/collection_browse_inventory.tab
3ff61c98cbed11e99690f76b5f6831b0  ./browse/ele_mom.pdf
fd014ead868211ac6421efd1633c1a33  ./browse/ele_mom_browse.xml
2e47917704231a1f8bd2d3a66e5588b3  ./data/collection_data.xml
2cef640ecac021083ef1fe3f03f4c6f6  ./data/collection_data_inventory.tab
2b555c42a7e7b4981407c9a824237f4a  ./data/ele_mom.tab
117982301d1b009b842104b134d86d3d  ./data/ele_mom_tblchar.xml
```

Since the checksum file is external to the package, and hence external to the archive, there is no requirement to provide a separate PDS4 XML label for it.

13.1.3 The Delivery Manifest

The delivery manifest is an optional file provided with a transfer to, from, or within PDS. The purpose of the delivery manifest is to show what products are intended to be included in the delivery. It is analogous to a packing slip in a physical package; by comparing the delivery manifest to the package contents, the recipient can tell if a product is missing from the delivery or if it contains any unexpected product(s). The delivery manifest may be required or optional depending on the policy of the PDS discipline node receiving the delivery.

The delivery manifest maps product LIDVIDs to file names for each product label file in the package. If the delivery package is a zip file, tar file, etc., the manifest must contain entries for every label file in the package after the package has been unpacked. The delivery manifest is in the form of a Table_Character file, a two-field fixed-width table with one row for each of the products in the package. The first field gives the LIDVID of each product in the package. The second field gives the name of the product's label file relative to the root directory of the package. Each record is delimited by an ASCII carriage-return line-feed pair.

The example below shows the structure of a delivery manifest:

```
urn:nasa:pds:example.dph.sample:browse:collection::1.0 ./browse/collection.xml
urn:nasa:pds:example.dph.sample:browse:ele_mon::1.0    ./browse/ele_mom_browse.xml
urn:nasa:pds:example.dph.sample:data:collection::1.0   ./data/collection.xml
urn:nasa:pds:example.dph.sample:data:ele_mon::1.0      ./data/ele_mom_data.xml
```

Since the delivery manifest is external to the package, and hence external to the archive, there is no requirement to provide a separate PDS4 XML label for it.

13.1.4 The Difference Between A Delivery Manifest and A Checksum File

The delivery manifest and the checksum file are not the same, and one file cannot serve both purposes. A delivery manifest has an entry for every *label* file in the delivery. It tells the recipient whether the delivery is complete; that is, whether it contains exactly what the sender intended to send. A checksum file has an entry for *every file* in the delivery – labels, data, and all. It tells the recipient whether any file in the delivery is corrupted. The delivery manifest may

or may not be required by the PDS node receiving the delivery. The checksum file is always required.

13.2 Transfer Procedures

You and your PDS node representative should agree on the procedure for delivering your archive to PDS following successful lien resolution. If your archive consists of a single delivery, then there may be nothing more for you to do if the PDS node already has the complete archive. Check with your node representative to be sure. On the other hand, if you will be delivering the archive incrementally, a more formal procedure should be put in place.

13.2.1 Logistics

When planning a delivery, both sides should have the following information.

- What is the format of the delivery package (e.g., a zip file)?
- What is the delivery mechanism (e.g., FTP "push" by provider to recipient's server)?
- What URL, account and password will be used to make the transfer?
- Is a delivery manifest required?
- What handshaking methods will be used? For example,
    - Notification of delivery
    - Notification of receipt
    - Notification of checksum verification, manifest verification, and validation status

If the delivery contains any damaged files, or is missing any files, it's the provider's job to replace them with intact ones, either individually or by repeating the whole delivery, as agreed between the two parties. If unexpected files are present, the provider must tell the recipient what to do with them. If there are validation errors, the data provider is responsible for correcting the errors.

13.2.2 Delivery Checklist

The following discussion applies mainly to missions delivering data in scheduled installments, say every three months, over a long period of time. A mission usually involves several instrument teams delivering to several PDS nodes at the same time, and one node may be receiving data from several teams. With so many interfaces and moving parts, a formal delivery plan is necessary.

A good approach is for each pair of provider and recipient to agree on a delivery checklist to follow. Here is an example of such a checklist.

Table 13-1. Example of an Archive Delivery Checklist

Step	Task	Responsibility
1.	Agree on transfer method for delivery	Provider and recipient

2.	Establish staging area at provider's site	Provider
3.	Establish staging area at recipient's site	Recipient
4.	Determine destination address, account and password	Provider and recipient
5.	Determine whether delivery manifest is required	Recipient
6.	Create delivery package, checksum file, and possibly delivery manifest in provider's staging area	Provider
7.	Notify recipient that delivery is ready for transfer	Provider
8.	Transfer the package to recipient's staging area	Provider
9.	Notify recipient that package has been delivered	Provider
10.	Notify provider that package has been received	Recipient
11.	Verify checksums and report to provider	Recipient
12.	Verify delivery manifest, if any, and report to provider	Recipient
13.	Validate the delivery for PDS compliance and for compliance with previously reviewed documentation, and report to provider	Recipient

After the first few deliveries the procedure may become routine enough that an actual checklist is no longer needed; but the checksum files, delivery manifests, and reports should continue to be produced, and the recipient PDS node should maintain them for future reference.

Most mission archives are cumulative; that is, each delivery adds more data to existing collections in an existing bundle, rather than a new bundle being created each time. For the second and subsequent deliveries of such an archive, the PDS node will probably ask the data provider to deliver only new files and files that have changed from previous deliveries. There should be no need to re-deliver parts of the archive that have not changed. See Section 13.3 for guidance about delivering revisions of previously released data products.

A note on the terms *delivery* and *release*: Both PDS and missions often talk about data releases as milestones, and missions usually publish a release schedule. A release happens when data are made publicly available on PDS web sites. A delivery, on the other hand, happens a few weeks before the release date, when a data provider delivers a package to a PDS node. (The schedule will be agreed upon by the mission and PDS.) Be careful to use the right terms when discussing data deliveries and releases to avoid misunderstandings.

13.3 Revising Previously Released Products

It's not unusual for data products that have been released to PDS to be revised later by the data provider, especially for mission-generated products. A common reason for revising data products is that calibration data or procedures have improved. Occasionally revisions occur to correct errors that have been discovered in the data. PDS policy is to retain previous versions of the data for at least six months in a place accessible to users, unless the revisions have been made to correct major errors that render the previous versions unfit for use. This policy applies to archives that are accumulating from an ongoing mission; contents are considered to have *draft*

status as long as the mission is ongoing. If data products in a closed collection – one for which the mission is over and no new products are expected – are revised, the previous versions are considered to be *superseded*, and will not be made available for use except in special circumstances. See the PDS policies for draft data and superseded data on the *PDS Policies* page, https://pds.nasa.gov/policy/.

PDS nodes are careful to ensure that the data available online are the most current versions but that older versions are retained for the specified time period. As a data provider you can help make this happen by informing your receiving node when a delivery contains revised products, exactly which products have been revised, and what you think should be done with the previous versions.

13.3.1 What Should Be Changed in the Labels of Revised Data Products

In a **Product_Observational** label the following attributes should be updated to indicate that the product has been revised:

`<version_id>` in the Identification Area

`<creation_date_time>` in the File Area

`Modification_History` in the Identification Area

creation_date_time and **Modification_History** are optional but strongly recommended when changes are introduced; the first can be very useful in confirming the version change; the latter can provide information on what was changed and why. `<logical_identifier>` in the Identification Area should *not* change. If you think this is incorrect, then what you have may not be a revision but a new product.

The file names associated with the product do not have to change, but they may. It may be easier for the PDS node to keep track of revisions to products if the file names stay the same, so discuss this with your node representative.

13.3.2 Versioning of Products, Collections, and Bundles

Recall that every product, collection, and bundle has a unique LIDVID, and that the VID part represents a version number. When a data product is revised, its VID should be incremented. A VID has the form M.n where M and n are both integers. M denotes a major version number and n denotes a minor version number. The first time a product is delivered to PDS its VID should be 1.0. If the product is revised later, either the minor version number should be incremented by 1, or the major version number incremented by 1 and the minor number set back to 0. It's up to the data provider to decide whether a revision is major or minor. Note that changes to a data product label count as a revision, even if the data product itself has not changed.

The addition or deletion of any product in a collection means that the collection itself has been revised because the collection inventory table has changed; so the collection VID should be incremented. If a product is revised, then the collection inventory table may or may not change, depending on whether it lists the products by LIDVID or just LID. If by LIDVID, then the inventory table will have to be revised and therefore the collection VID will have to be incremented.

The same is true of bundles. If a collection is added or removed from a bundle, then the bundle inventory must change and therefore the bundle VID must be incremented. If the inventory lists the collections by LIDVID, and one of the collections is revised, then the inventory must be changed and the bundle VID incremented.

The *Standards Reference* [2] Section 6D.3 explains versioning in detail.

13.4 Making Products Available to Users

Once a delivery has been received, validated, and accepted by a PDS node, it is ready to be released. The PDS node will make it available online and announce it on the node's web site. The node representative will inform the responsible PDS Engineering Node data engineer, who will ensure that the data are also available from the main PDS web site via the search function on the PDS home page (https://pds.nasa.gov).

PDS maintains a subscription service to announce new data releases (https://pds.nasa.gov/tools/subscription_service/). Users may subscribe to receive email announcements about releases of selected data sets, documentation, and software. When a node representative informs the PDS data engineer of a new release, together they create an announcement to be sent to subscribers. The announcement may also be posted in media outlets likely to be seen by users, such as the Planetary Exploration Newsletter (http://planetarynews.org).

A mission or data provider may request that a release be posted and announced at a particular date and time, for example to coincide with the release of a journal publication about the data. PDS will comply with the request as closely as possible.

Appendices

Appendix A Acronyms and Abbreviations

AMMOS	Advanced Multi-Mission Operations System
APL	Applied Physics Laboratory
APPS	AMMOS-PDS Pipeline Service
ASCII	American Standard Code for Information Interchange
CR	Carriage Return
CSV	Comma-Separated Value
DIP	Dissemination Information Package
DN	Discipline Node (PDS)
DOI	Digital Object Identifier
DOM	Document Object Model
DPH	Data Provider's Handbook
DTD	Document Type Definition
EN	Engineering Node (PDS)
ESA	European Space Agency
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GSFC	Goddard Space Flight Center (NASA)
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IM	Information Model
ISO	International Organization for Standardization
ITAR	International Traffic in Arms Regulations
J2C	JPEG-2000 compression algorithm
JHU	Johns Hopkins University
JPEG	Joint Photographic Experts Group (compression algorithm)
JPEG2000	JPEG compression algorithm created in 2000 to supersede original JPEG
JPL	Jet Propulsion Laboratory
LACE	Label Creation and Editing (tool)
LDD	Local Data Dictionary

LDT	Label Design Tool
LF	Line Feed
LID	Logical Identifier
LIDVID	Versioned LID
LSB	Least Significant Byte First (storage order)
MAVEN	Mars Atmosphere and Volatile Evolution (mission)
MD5	Message Digest 5 (checksum format)
MSB	Most Significant Byte first (storage order)
NAIF	Navigation and Ancillary Information Node (PDS)
NASA	National Aeronautics and Space Administration
P	Primary (member)
PDF	Portable Document Format
PDF/A	Portable Document Format (archival)
PDS	Planetary Data System
PDS3	PDS version 3
PDS4	PDS version 4
PNG	Portable Network Graphics
PPI	Planetary Plasma Interactions (PDS DN)
PSA	Planetary Science Archive (European Space Agency)
S	Secondary (member)
SBN	Small Bodies Node (PDS)
SCH	Schematron
SIS	Software Interface Specification
SPICE	Spacecraft, Planet, Instrument, Camera Matrix, and Event (historic acronym for system of storing planetary navigation and other ancillary information)
TIFF	Tagged Image File Format
UCD	User Centered Design
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Coordinated Universal Time
UTF-8	Unicode Transformation Format 8 (character set)
VID	Version Identifier

XAE	XML-aware editor
XML	eXtensible Markup Language
Xpath	XML Path Language
XSD	XML Schema Definition

Appendix B Archive Lessons Learned from Past Missions

These comments come from missions which have recently archived with PDS4 – lessons learned, things they wish they had done, or things they did do that improved their archiving experience. This appendix is mentioned in Section 1.3, Reader Preparation, Section 3, Checklist for Creating a PDS Archive, and Section 4.3.2, Directory and File Naming.

- Think of archiving as an integral part of the mission, not an additional activity.
- Initiate contact between PDS and the mission early; face-to-face discussion is particularly useful.
- Design an archive schedule with mission activities in mind. (Setting SIS due dates within a week of the target date for integrating instruments with the spacecraft is not a good idea!)
- Identify archive conventions which the mission should establish early. This is as much about coordinating among groups within the mission as it is about coordinating the mission with PDS:
    - Determine what data products will be archived at each processing level (raw, calibrated, high level)
    - Establish the archive organization (define bundles and collections and how products will be assigned to them)
    - Agree on a versioning scheme for the components of the archive (what represents a new version of a data product or a document, and how will the versions be represented?)
    - Establish naming conventions for bundles, collections, products, directories, and files.
    - Agree on a set of common time formats (for example, decide whether to use YYYY-MM-DD or YYYY-DDD for dates)
    - Agree on a set of common data formats (for example, can most tabular data be archived in fixed-width ASCII tables?)

Appendix C How To Select the Right Class for a Product

This appendix continues the discussion in Section 6.2.1 about identifying the type of product to be described by a PDS label. The questions below outline the decision process used to select the appropriate product class when the choice is not obvious. For simplicity this outline assumes there is one data object in the product. The outline is shown schematically in Figure C-1.

1. Is the object digital, a bundle, a collection, or an update to an existing product? If none of these, use **Product_Context**.
2. Does the object contain PDS operational data? If so, use one of several classes for operational products defined in Chapter 3 of the *Data Dictionary* [3] (these are not, strictly speaking, science data). Operational products contain behind-the-scenes data such as class and attribute definitions, and are probably not of interest to science data providers.
3. Is this an aggregation product? If so, choose between **Product_Collection** and **Product_Bundle**. **Product_Collection** contains an Inventory table listing basic products. **Product_Bundle** contains a list of member collections.
4. Is this a product that supports the archive rather than being data per se? That is …
    a. Is this a ZIP compressed file? If so, use **Product_Zipped**.
    b. Is this an XML schema and/or Schematron file? If so, use **Product_XML_Schema**.
    c. If (4) but neither (a) nor (b) and it is ASCII_Short_String_Collapsed with <CR><LF> record delimiters, then use **Product_File_Text**.
    d. If (4) but not (a), (b), or (c) work with the consulting DN to determine the appropriate product type.
5. Is this a SPICE kernel? If so, use **Product_SPICE_Kernel**.
6. Is this solely a finding aid for humans? If so,
    a. Is this used in an image index? If so, use **Product_Thumbnail**.
    b. If not to be used in an image index, use **Product_Browse**.
7. Is this intended primarily to be read by humans? If so, use **Product_Document**.
8. Does the digital object contain observational data?
    a. If it contains observational data, does it have one of the following PDS4 structures: **Array**, **Encoded_Header**, **Header**, **Stream_Text**, **Table_Binary**, **Table_Character**, or **Table_Delimited**?
        i. If it has one of these structures and the processing level is “raw”, use **Product_Observational**. (See Table 2-1 for processing level definitions.)
        ii. If it has one of these structures, is not “raw”, and its function is primarily to support other products in the archive, use **Product_Ancillary**.

iii. If it has one of these structures, is not “raw”, and its function is not primarily to support other products in the archive, use **Product_Observational**.

iv. If it does not have one of these structures but it is in an original format (such as from the observing system), use **Product_Native** (see Section 9E for special requirements)

v. If it does not have one of these structures and is not in an original observing system format, use **Product_Ancillary**.

b. If it does not contain observational data, use **Product_Ancillary**.

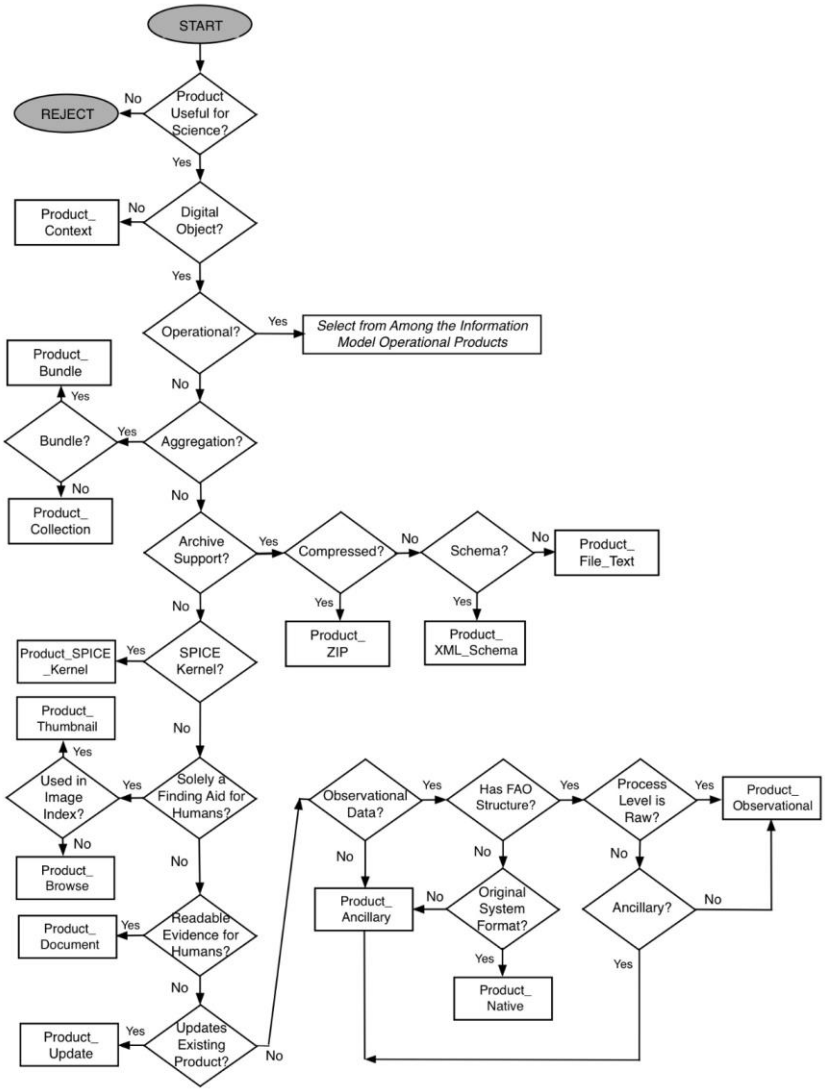

Figure C-1. Decision Process For Determining What Product Class to Use

Appendix D XML Schema Basics

This appendix gives a very basic introduction to XML and XML Schema for the novice user, without getting into PDS-specific details. Section 6, Designing Labels, refers to this appendix.

XML (https://www.w3.org/TR/xml/) is a language for writing documents that are intended to be parsed by software. For example, HTML is a specific implementation of XML that is used for writing web pages. PDS labels are written in XML.

XML alone is not sufficient for recording metadata for science products. To organize the metadata it is necessary to impose a structure. Without a structure, you would not know what to put in your PDS XML labels. You would have to make up attributes and classes and validation rules, and no two providers would do it the same way. XML Schema (https://www.w3.org/TR/xmlschema11-1/) is a standard for defining a structure for XML documents.

Schematron (http://schematron.com/) is another standard that PDS uses to ensure quality metadata in labels. It is a way to write validation rules for XML documents when the rules are too complicated to be expressed using XML Schema alone. This is discussed in Appendix J.

D.1 Reading XML Files

At first glance an XML file can appear to be dense and rather cryptic text sprinkled with angle brackets. It helps to view the file in an editor that understands XML and can provide color-coding and other embellishments to make components of the file stand out. See Section 6.4.1, XML-Aware Editors.

You may be familiar with PDS3 labels, which used a `keyword = value` format to record metadata, as in `FILE_NAME = "MYFILE.DAT"`. XML uses *tags*:

```
<file_name>myfile.dat</file_name>
```

In this example `<file_name>` is the tag, which is analogous to the keyword, and `myfile.dat` is the *content*. Every tag includes its angle brackets, and every tag must have a matching *closing tag*, which includes a forward slash. The full line — tag, content, and closing tag — is called an *XML element*. Tags can be nested.

White space outside of tags is ignored, so you can add blank lines, spaces and line endings however you like to make the file easier to read.

Angle brackets (< and >) and the ampersand character (&) have special meaning in XML, so they should not be used in your label except to delimit tags and to indicate special characters. (For example, ` ` indicates a non-breaking space. You won't see many of these special characters in PDS labels.)

Comments in XML start with `<!--` and end with `-->`.

The *PDS Small Bodies Node Wiki* (see Section 1.4.4, http://sbndev.astro.umd.edu/wiki/SBN_PDS4_Wiki) has a more complete overview of XML as it applies to PDS4.

D.2 Using XML Schema

In order for any XML document, including a PDS4 label, to meet the XML standard, it must be both *well-formed* and *valid*. A well-formed XML document must have correct XML syntax; a valid XML document must conform to the rules of an XML schema document (which you'll find in a file with the extension .xsd) and, if applicable, an XML Schematron document (extension .sch). Under PDS4, schemas provide the rules governing the structure and some of the content of each XML class, and Schematron documents further constrain the contents of those classes and attributes within classes.

The PDS4 common schema specifies the content of a PDS label, the order of the classes and attributes, which ones are required and which are optional, and what kinds of values are expected for each attribute.

Figure D-1 illustrates the PDS4 label components that tell label-parsing software what schemas and Schematron documents apply to the label. Every label begins with an XML prolog that identifies it as an XML document and identifies any Schematron documents that apply to it (the blue text in the figure). The rest of the label is enclosed between a pair of XML tags corresponding to the class of the product being described, such as Product_Observational. The opening tag includes information about the namespaces used in the label (red text) and the locations of the XML schemas that go with those namespaces (green text). These components are explained in more detail in Appendix F.3, Editing the XML Prolog and Root Tag.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
    schematypens="http://purl.oclc.org/dsdl/schematron"?>
<?xml-model href="http://pds.nasa.gov/pds4/disp/v1/PDS4_DISP_1301.sch"
    schematypens="http://purl.oclc.org/dsdl/schematron"?>
<?xml-model href="http://pds.nasa.gov/pds4/sp/v1/PDS4_SP_1100.sch"
    schematypens="http://purl.oclc.org/dsdl/schematron"?>

<Product_Observational
     xmlns="http://pds.nasa.gov/pds4/pds/v1"
     xmlns:pds="http://pds.nasa.gov/pds4/pds/v1"
     xmlns:disp="http://pds.nasa.gov/pds4/disp/v1"
     xmlns:sp="http://pds.nasa.gov/pds4/sp/v1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                         http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd
                         http://pds.nasa.gov/pds4/disp/v1
                         http://pds.nasa.gov/pds4/disp/v1/PDS4_DISP_1301.xsd
                         http://pds.nasa.gov/pds4/sp/v1
                         http://pds.nasa.gov/pds4/sp/v1/PDS4_SP_1100.xsd">

     [many lines omitted]

</Product_Observational>
```

Figure D-1. Schema and Schematron References in a PDS4 Label

Appendix E Creating an Initial Draft Label

This appendix lists the steps to be used in creating a draft label that can then be modified to become a label template, as discussed in Section 6.4. This discussion focuses on specific instructions for the Eclipse XML editor and the oXygen XML editor.

E.1 Appropriation

The simplest way to create a draft label is to copy the label from a product that is similar to the one you are planning to label. Your consulting DN data engineer may be able to provide one. If you have in hand a label that describes the kind of product you will be archiving, and the label is fairly recent (the latest PDS4 Information Model, if possible), then you can skip the rest of this appendix and go to Appendix F, XML Label Editing, to learn how to modify the label to make your template.

E.2 Eclipse XML Editor

The Small Bodies Node wiki includes instructions on generating new label files from the PDS4 common schema using the Eclipse XML editor. Go to:

http://sbndev.astro.umd.edu/wiki/Eclipse:_Creating_a_New_XML_File_from_an_XSD_Schema_File

The SBN wiki pages also cover editing of the XML Declaration and Root Tag, so if you follow those instructions you can skip the corresponding parts of Appendix F in this handbook.

E.3 oXygen XML Editor

This set of steps is based on the oXygen 17.0 XML editor (other versions of oXygen should perform similarly). It produces a label from the PDS4 common schema (the xsd file); the label has all levels and all options. You must then remove options you don't want by subsequent editing. XML elements for attributes are created with empty values by default, but you can select an option which inserts values that are either valid or not, your choice. You then need to edit those by inserting legitimate values for your labels.

Step 1: Download a copy of the current common schema (xsd)

a. The current version of the PDS4 common schema (xsd) can always be found at the PDS4 Schema web page. Go to: https://pds.nasa.gov/pds4/schema/released/.

b. Click to download

   - The current Schematron file: PDS4_PDS_nnnn.sch
   - The current common schema file: PDS4_PDS_nnnn.xsd

   The third file, PDS4_PDS_nnnn.xml, is a PDS4 label that describes the other two. You may wish to download it also, but it is not needed for the exercise we are going to do now.

c. Save the files to a working directory or folder.

Step 2: Open the common-schema in an XML-aware Editor

a. Using the oXygen XML editor, open the common schema (e.g. PDS4_PDS_1700.xsd) in your working directory.

b. Pull down the **Tools** menu from the top menu bar and select **Generate Sample XML Files …** You will get a window like the one shown in Figure D-3.

c. From the three tabs at the top, select the **Schema** tab.

d. The **URL** entry will be blank. Use the folder icon at the right to select the local copy of the common schema (the xsd file you just downloaded). This will also populate the **Namespace** with the appropriate value from the xsd file.

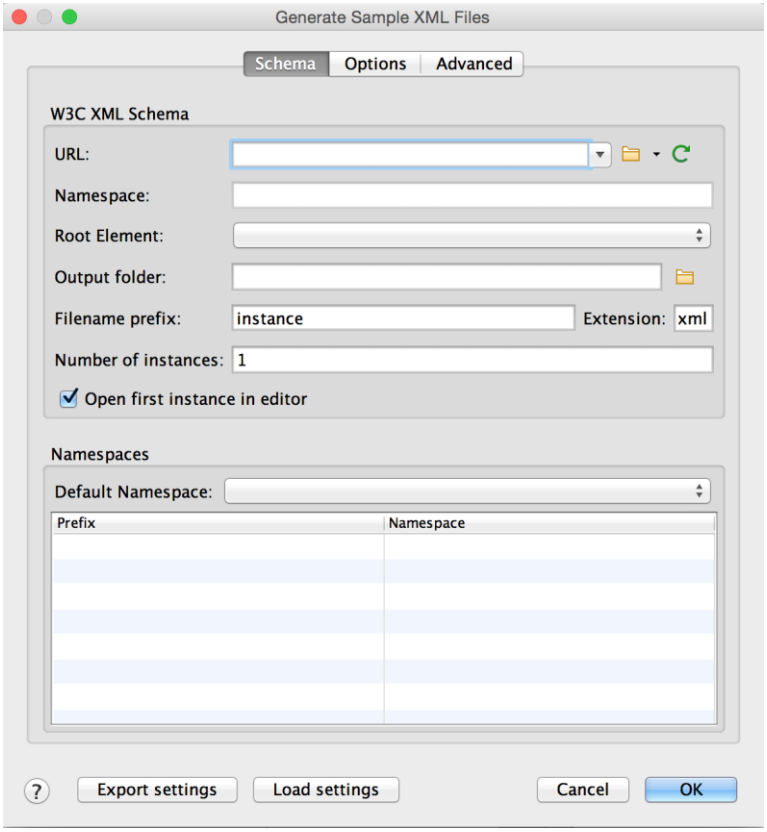

Figure E-1. Using oXygen to Generate an XML Label, part 1

e. Use the down arrow and scroll bar next to the **Root Element** box to select the product type you want — for example, Product_Observational.

f. The **Output folder** box is filled in with the path to your common schema file. Change this to the path to your working directory, if it is different.

g. In the **Filename prefix** box, enter the base file name for your label file. As noted at the right, it will be given the extension *.xml.

Step 3: Select Options

a. Do not select **OK** yet. From the three tabs at the top, select the **Options** tab. You will get a window like the one shown in Figure D-4

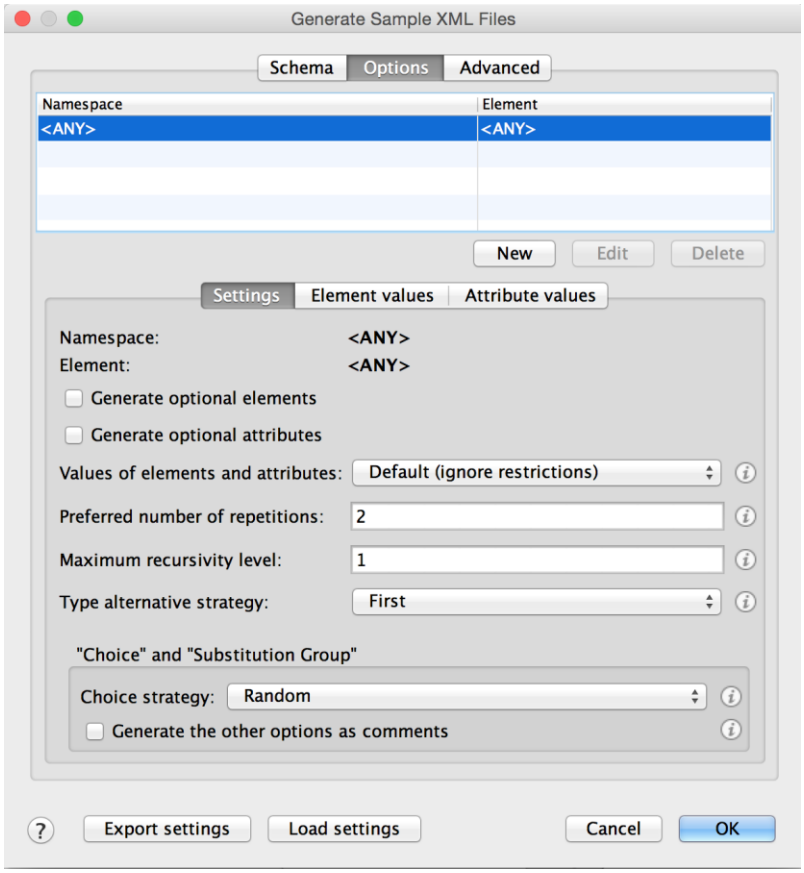

Figure E-2. Using oXygen to Generate an XML Label, part 2

b. Check the boxes for **Generate optional elements** and **Generate optional attributes**.

c. For the **Value of elements and attributes** you have three choices:

None

Default (ignore restrictions)

Random (apply restrictions)

You should select **None**; this means that all of the attributes in the label will have empty values. The empty values will be flagged by the real time validation function of the oXygen XML editor; you can insert values and clear the error flags when you edit the label later.

If you select **Default (ignore restrictions)**, meaningless values will be inserted. The real-time oXygen validation may not report errors. If you select **Random (apply restrictions)**, plausible values will be inserted. In both cases, you will have to check the label *very* carefully to prevent bad values from leaking into the final labels. It is simpler and safer to select **None** and work through the label, clearing flagged errors (empty values).

d. Set **Preferred number of repetitions** to 1. If you need repetitions, you can add them later when you edit the label.

e. In the **Type alternative strategy** and **Choice strategy** boxes, select **First**. If you select **Random**, the results may be unpredictable. **Random** is useful if you're simply interested in surveying example labels, but when generating a label that will become a label template, you want predictable results.

f. Check **Generate the other options as comments**. This fleshes out the label with all possible options. Removing the comment notation is not difficult; but removing the large number of unwanted options is the primary disadvantage of this method.

Step 4: Create the label (XML) document

a. Click OK at the bottom right of the window.

b. The label will auto-display in oXygen.

c. The label has everything you need (and quite a bit more). Options included as comments will be set off by <!-- at the left margin (and by --> after the corresponding end tag, which is harder to spot). You can accept the ones you want by removing the special punctuation; you can remove the ones you don't want by deleting the appropriate rows.

d. Save the label file to your working directory; it will have the unique file name you specified on the Schema page (Figure E-1).

Step 5: Ensure the label is valid (or not)

a. Your initial label is very likely not valid; little red boxes (or lines) in the right scroll bar indicate where errors exist. Most, if not all, of the errors in the initial label result from missing values.

b. As you proceed with editing the label (Appendix F of this handbook), you will clear the errors, and the red boxes will disappear. Note that the editor may not flag all of the errors initially. As you clear errors early in the file, red boxes will appear later in the file for errors that have been newly identified.

c. Once a full set of values has been specified, the label should become valid, indicated by a green box at the top of the right-hand scroll bar.

Now that you have an initial draft of your label, you are ready to modify it to create a label template for your data products. A label template is just a label that has some placeholders instead of actual values. The placeholders will be filled in later by software.

Appendix F discusses editing the draft label to create a label template.

Appendix F XML Label Editing

This appendix goes into the details of using the oXygen XML editor to modify an initial draft label. If you do not already have an initial draft label, see Appendix E. For an overview of PDS4 XML labels, see Section 6.1.

The following instructions walk you through the process of editing a draft label to create a single label for a particular product, and then a label template that can be used to generate labels for many products. Your consulting node representative can help you generate a label template for each type of product that will appear in your archive. If you have several different product types with the same structure — for example, different ASCII tables for solar flux and magnetic field — you will probably need separate templates for each.

The PDS Small Bodies Node (SBN) maintains a good online resource for step-by-step instructions in filling out PDS labels at the SBN PDS4 Wiki, on the page http://sbndev.astro.umd.edu/wiki/PDS4_Product_Labels,_Step_by_Step.

F.1 Getting Started with an Example

In the *DPH Examples* at https://pds.nasa.gov/pds4/doc/examples/ you'll find the file **Table_Character_draft1.xml**. This is an initial draft label that was generated in oXygen using the instructions in Appendix E. You may wish to open this file in your own XML-aware editor to follow along as we go over the ways to modify a draft label. When you first open it, the contents may be confusing; refer to the high-level label diagram in Figure 6-1 to get your bearings.

Although the `<XML Prolog>` and the `<Product_Observational>` root tag appear at the beginning of the label, we will discuss editing of the body of the label first. We will return to the XML Prolog and Root Tag later.

As you go through the following sections, keep in mind that this is a tutorial intended to show you the mechanics of editing a PDS4 label, using a simple example. It is not intended to be a comprehensive explanation of every class and attribute that can appear in a label. If you are in doubt about what classes, attributes, and values to include when designing your own label, consult the online resources listed in Section 1.4 and your PDS node representative.

F.2 Editing the Body of the Label

We're going to modify the draft label **Table_Character_draft1.xml** to make it describe a reflectance spectrum that is stored as an ASCII text table with three columns (*fields*) and 224 rows (*records*). Figure F-1 shows a few lines from the beginning and end of this table.

Commented [SS7]: I don't have Eclipse so can't write instructions for it.

Commented [EAG8]: It seems to me that it is important to add the model statement early, which has to be done manually, so that schematron rules are checked. Otherwise it may appear like the label is valid, but it might not be once the rules are checked.

Commented [SS9R8]: I thought the XML prolog and root section was just too overwhelming for the new user, so I started with something more familiar, describing a data file. But I added F.2.1 to address your concern, and it says you can do the other section first and then come back here.

```
     320.0   0.05039   0.01451
     330.0   0.05898   0.01123
     340.0   0.06849   0.01162
     350.0   0.07776   0.00851
     360.0   0.08820   0.00314
…
    2520.0   0.32924   0.00205
    2530.0   0.32996   0.00385
    2540.0   0.32930   0.00210
    2550.0   0.33209   0.00437
```

Figure F-1. Example of Table_Character data

The table has fixed-width fields, which means it should be described by the **Product_Observational** subclass **Table_Character** (Section 6.2.2.1). First we'll create a valid label that describes a single table, and then we'll turn it into a label template that can be used to generate labels for a set of tables.

We're going to start by modifying the `<File_Area_Observational>` tag, and then we'll work on the other areas in the body of the label. We're doing things in this order to help those who are completely new to editing PDS4 labels; it's easier to start with what is familiar – the contents of your data files – and move from there to less familiar areas.

F.2.1 A Note on Validation

Section 6.4.1 mentioned that using an XML-aware editor is helpful because it will instantly validate your label based on whether it is compliant with its controlling schema(s). As we are working on the body of the label first, the editor may not correctly validate the label yet; we may need to modify the schema and Schematron files specified in the XML prolog and root tag before we can rely on the editor's validation. That comes in Section F.3. If you would rather work on the XML prolog and root tag first, go on to Section F.3, and then return here.

When the XML prolog, the root tag, and the body of the label have all been modified correctly, the XML-aware editor should show the label as valid. In oXygen this is shown as a green square in the upper right corner of the editing window.

One more comment about validation: The XML-aware editor can only tell you whether a label is valid according to its associated schemas and Schematron files. It cannot tell you whether the label correctly describes its associate data product; you could create a valid label even in the absence of any data. Other tools must be used to do a more complete validation. See Section 11, Archive Validation.

F.2.2 Modifying File_Area_Observational

To start, we will close up all but the highest level tags so that we can see an "outline" of the label. (Look in your editor's left margin for an arrow or other symbol to click that will open and close a tag.) The result will look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Observation_Area>
    <Reference_List>
    <File_Area_Observational>
    <File_Area_Observational_Supplemental>
</Product_Observational>
```

F.2.2.1 Removing Unwanted Product Types

Opening up the `<File_Area_Observational>` tag reveals a `<File>` tag, an `<Array>` tag, and all the other possible subclasses of **Product_Observational**, which are given inside comment delimiters. The draft label appears this way because of the choice we made when creating it in step 3.f of Appendix E, **Generate the other options as comments**. The `<Array>` tag is not inside comment delimiters just because it happens to come first alphabetically. By closing up all the comment fields, the label now appears like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Observation_Area>
    <Reference_List>
    <File_Area_Observational>
        <File>
            <file_name></file_name>
            <local_identifier></local_identifier>
            <creation_date_time></creation_date_time>
            <file_size unit=""></file_size>
            <records></records>
            <md5_checksum></md5_checksum>
            <comment></comment>
        </File>
        <Array>
<!--        <Array_1D>
<!--        <Array_2D>
<!--        <Array_2D_Image>
<!--        <Array_2D_Map>
<!--        <Array_2D_Spectrum>
<!--        <Array_3D>
<!--        <Array_3D_Image>
<!--        <Array_3D_Movie>
<!--        <Array_3D_Spectrum>
<!--        <Encoded_Header>
<!--        <Header>
<!--        <Stream_Text>
<!--        <Table_Binary>
<!--        <Table_Character>
```

```
<!--        <Table_Delimited>
    </File_Area_Observational>
    <File_Area_Observational_Supplemental>
</Product_Observational>
```

We are creating a label for a **Table_Character** product, so we will now delete all the other types of products from this part of the label, and remove the comment delimiters from around **Table_Character**. Now the label looks like this, with the `<Table_Character>` tag closed up:

```
<?xml version="1.0" encoding="UTF-8"?>
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Observation_Area>
    <Reference_List>
    <File_Area_Observational>
        <File>
            <file_name></file_name>
            <local_identifier></local_identifier>
            <creation_date_time></creation_date_time>
            <file_size unit=""></file_size>
            <records></records>
            <md5_checksum></md5_checksum>
            <comment></comment>
        </File>
        <Table_Character>
    </File_Area_Observational>
    <File_Area_Observational_Supplemental>
</Product_Observational>
```

F.2.2.2 Removing Optional Classes and Attributes

Opening up the `<Table Character>` tag reveals all the possible classes and attributes that apply to **Table_Character** observational products. The label appears this way because we chose **Generate optional elements** and **Generate optional attributes** in step 3.b in Appendix E.

We can now delete any of the optional parts of the label that we don't need. A class or attribute is optional if its definition in the data dictionary specifies `minOccurs="0"` (minimum number of occurrences is 0). In oXygen you can see the definition of a class or attribute by right-clicking on its name and choosing **Show Definition**. This opens the dictionary in the editor and highlights the definition. For example, by right-clicking on the tag `<Uniformly_Sampled>` inside the `<Table_Character>` tag, we see that its specification is

```
<xs:element name="Uniformly_Sampled" type="pds:Uniformly_Sampled" minOccurs="0"
maxOccurs="1"> </xs:element>
```

This means the **Uniformly Sampled** class is not required (minOccurs = "0"), and may appear no more than once in a label (maxOccurs="1"). We will delete it from our label because it doesn't apply to the product we are labeling. We'll also delete **local_identifier**, another optional attribute. We'll keep **name** and **description**, even though they are optional. The other attributes and the **Record_Character** class are required.

Now the <Table_Character> tag in the label looks like this, with the <Field_Character> tag closed up for the moment:

```
<Table_Character>
    <name></name>
    <offset unit=""></offset>
    <records></records>
    <description></description>
    <record_delimiter></record_delimiter>
    <Record_Character>
        <fields></fields>
        <groups></groups>
        <record_length unit=""></record_length>
        <Field_Character>
    </Record_Character>
</Table_Character>
```

F.2.2.3 Setting the Values of Attributes

We can now start filling in some values in the label. The values are simply inserted between the starting and ending tags of the attributes. These will be values that apply to a single data product. Later we will discuss how to turn this label into a label template by inserting placeholders for those values that vary from one product to the next.

Here is the <Table_Character> tag with values filled in (again, with <Field_Character> closed up for now):

```
<Table_Character>
    <name>Reflectance Spectrum</name>
    <offset unit="byte">0</offset>
    <records>224</records>
    <description>This table contains a reflectance spectrum with
        fields Wavelength, Reflectance, and Error.</description>
    <record_delimiter>Carriage-Return Line-Feed</record_delimiter>
    <Record_Character>
        <fields>3</fields>
        <groups>0</groups>
        <record_length unit="byte">31</record_length>
        <Field_Character>
    </Record_Character>
</Table_Character>
```

F.2.2.4 Choosing Values From an Enumerated List

Some attributes are defined with a limited set of permissible values, called an enumerated list. One such attribute is <record_delimiter> in the above example. To see the values to choose from, you may open the *PDS4 Information Model* [1] in a web browser at https://pds.nasa.gov/pds4/doc/im/current/ and use the browser's search tool to look up "record_delimiter". It appears in several places, so make sure you are looking at the specification for record_delimiter in the **Table_Character** class – it's in Section 9.37.

F.2.2.5 Repeating a Class or Attribute

As our table has three fields, or columns, we have set the value of <fields> to 3 in the <Record_Character> tag. This tag already includes a <Field_Character> tag that we will use to describe one field, but we will need two more for the second and third fields. We can simply copy and paste additional instances of the <Field_Character> tag. To check whether there is a limit on the number of fields that can be defined in a record, we can right-click on <Field_Character> to see its definition:

```
<xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="Field_Character" type="pds:Field_Character"> </xs:element>
    <xs:element name="Group_Field_Character" type="pds:Group_Field_Character">
     </xs:element>
</xs:choice>
```

This is a little more complicated than the first definition we looked up. The <xs:choice> tag means that the **Record_Character** class is required to have at least one **Field_Character** subclass or **Group_Field_Character** subclass, and it may have as many of them as needed (maxOccurs="unbounded").

Here is the <Record_Character> tag in our example label now with three <Field_Character> tags expanded and filled in.

```
<Record_Character>
    <fields>3</fields>
    <groups>0</groups>
    <record_length unit="byte">31</record_length>
    <Field_Character>
        <name>Wavelength</name>
        <field_number>1</field_number>
        <field_location unit="byte">1</field_location>
        <data_type>ASCII_Real</data_type>
        <field_length unit="byte">9</field_length>
    </Field_Character>
    <Field_Character>
        <name>Reflectance</name>
        <field_number>2</field_number>
        <field_location unit="byte">10</field_location>
        <data_type>ASCII_Real</data_type>
        <field_length unit="byte">10</field_length>
    </Field_Character>
    <Field_Character>
        <name>Error</name>
        <field_number>3</field_number>
        <field_location unit="byte">20</field_location>
        <data_type>ASCII_Real</data_type>
        <field_length unit="byte">10</field_length>
    </Field_Character>
</Record_Character>
```

The attributes **field_location** and **field_length** are important. They give the starting byte and number of bytes respectively for each field in the record, with the first byte in the record numbered byte 1. The **field_location** and **field_length** values do not count any spaces or delimiters that may appear between fields in a fixed-width record, nor do they count any

quotation marks that surround field values.[12] To make things more complicated, a table may have groups of fields that repeat, which are described using the **Group_Field_Character** class (or **Group_Field_Delimited** or **Group_Field_Binary** classes for **Table_Delimited** and **Table_Binary** tables). See Appendix H, Counting Fields and Groups in a Nested Structure, for help in determining the right values for **field_location** and **field_length** in a table that contains groups.

F.2.2.6 Adding an Optional Class or Attribute

Commented [SS10]: Is there a clever way to do this using Oxygen?

In the `<Field_Character>` tags above we left out several of the optional attributes. Suppose now we decide that we want to include some of them after all. In particular, for the Wavelength field we want to specify that the data are in units of nanometers. It's important to remember that *order matters*; the attributes and classes must appear in the label in the same order in which they are specified in the schema where they are defined. The online version of the *Information Model* [1] is not helpful for this purpose because it gives the attributes of a class in alphabetical order. To see them in schema-defined order we must look in the *PDS4 Data Dictionary* [3], https://pds.nasa.gov/pds4/doc/dd/current/.

Searching in that document for "Field_Character", we see in its definition that the attribute **unit** should come after **field_length** in our label (as we have omitted the optional attribute **field_format** that would come between them). If we try to put it somewhere else in the `<Field_Character>` tag, the editor will flag it as an error.

F.2.2.7 Choosing a Value for the unit Attribute

Now that we have restored the unit attribute, what value should it have? Nanometers or nm? Unfortunately there is no easy way to look up an enumerated list of values for units because there are so many of them and they can be combined for complex measurements. In general, PDS uses SI units and abbreviations. In our simple case, either `<unit>nm</unit>` or `<unit>nanometer</unit>` will do. (Note that the unit name is singular, not plural.) See the *PDS Standards Reference* [2] Section 7, Units, and ask your consulting node if you need help determining the right unit name. The *Small Bodies Node PDS4 Wiki* has a section on Units of Measure at http://sbndev.astro.umd.edu/wiki/Units_of_Measure.

So now the wavelength field has this definition:

```
<Field_Character>
    <name>Wavelength</name>
    <field_number>1</field_number>
    <field_location unit="byte">1</field_location>
    <data_type>ASCII_Real</data_type>
    <field_length unit="byte">9</field_length>
    <unit>nm</unit>
</Field_Character>
```

[12] Different attributes are used to describe variable-width records; see the *Information Model* [1] sections on the **Table_Delimited**, **Record_Delimited**, and **Field_Delimited** classes. For more information about tabular data products, see the *Standards Reference* [2] Section 4B, Table Base.

You may notice another kind of unit specification in the example: `<field_location unit="byte">1</field_location>` and `<field_length unit="byte">9</field_length>`. These attributes refer to the field's starting byte and number of bytes in the record. In this case **byte** is the unit for a label attribute, so it appears as part of the attribute tag, as opposed to **nm**, which is the unit for data in the table.

F.2.2.8 Setting Values for Special Constants

One of the subclasses we left out of the **Field_Character** class is **Special_Constants**. A special constant is a value that may occur in a data product to indicate a special case, such as unknown, invalid, or missing data. Look in the *PDS4 Data Dictionary* [3] under **Special_Constants** to see the complete list. In our example, let's suppose that some of the reflectance values are missing, and that in the table these are given as -999.9 (choosing a value that can't be a real data value). We will put back the **Special_Constants** class in the Reflectance field definition to indicate that a data value of -999.9 means the reflectance at that wavelength is missing.

Looking at the *Data Dictionary* entry for **Field_Character** as we did before for **unit**, we can see that the **Special_Constants** class should come after the **unit** attribute in our label, as we have omitted the intervening optional attributes of **scaling_factor**, **value_offset**, and **description**. Then by clicking on "Special_Constants" in the far right column of the **Field_Character** entry, we go to the definition of Special Constants where the different kinds of constants are listed. The attribute we want is **missing_constant**. Here, then, is our updated `<Field_Character>` tag:

```
<Field_Character>
    <name>Reflectance</name>
    <field_number>2</field_number>
    <field_location unit="byte">10</field_location>
    <data_type>ASCII_Real</data_type>
    <field_length unit="byte">10</field_length>
    <Special_Constants>
        <missing_constant>-999.9</missing_constant>
    </Special_Constants>
</Field_Character>
```

Note that the value that is chosen for a special constant must be the right data type for the field. That is, we chose -999.9 instead of -999 because the field is defined as ASCII_Real.

F.2.2.9 The File Tag

Now that our label describes the data file with the `<Table_Character>` tag, let's fill in the remainder of the **File_Area_Observational** area of the label. Just above the `<Table_Character>` tag is the `<File>` tag, where we specify the file name and other characteristics. The only required attribute is **file_name**, but we'll include **creation_date_time** and **records** as well.

```
<File>
    <file_name>Table_Character_Example.tab</file_name>
    <creation_date_time>2017-03-24T12:01</creation_date_time>
    <records>224</records>
</File>
```

The **creation_date_time** must be given in the PDS standard date-time format, as specified in the *Standards Reference* [2] Section 5A.2.

F.2.3 Modifying File_Area_Observational_Supplemental

The **File_Area_Observational_Supplemental** part of the label is used to describe any supplemental data that may accompany a **Product_Observational** data product. For example, an image may be accompanied by a histogram or other statistics. A table may be accompanied by a figure showing the data plotted on a graph. The supplemental data may be in a separate file from the **Product_Observational** data, or it may be included in the same file.

Let's assume our example table of reflectance data comes with a plot of the data in a separate file, a PNG image.

Modifying **File_Area_Observational_Supplemental** is done in much the same way as **File_Area_Observational**. You can follow the same directions in Section F.2.2, deleting the unwanted product types and removing the comment delimiters from the desired product type. In the case of our PNG image, the desired product type is **Encoded _Image**. (Refer to Section 6.2 and Appendix C for help choosing the product type.) Here is a completed `<File_Area_Observational_Supplemental>` tag for our example:

```
<File_Area_Observational_Supplemental>
    <File>
        <file_name>Table_Character_Example_Plot.png</file_name>
    </File>
    <Encoded_Image>
        <offset unit="byte">0</offset>
        <encoding_standard_id>PNG</encoding_standard_id>
        <description>This image is a plot of the reflectance data.</description>
    </Encoded_Image>
</File_Area_Observational_Supplemental>
```

F.2.4 Modifying Identification_Area

To continue with our example label, let's now open up the `<Identification_Area>` tag near the top of the label. With the subclasses still closed up, the tag looks like this:

```
<Identification_Area>
    <logical_identifier></logical_identifier>
    <version_id></version_id>
    <title></title>
    <information_model_version></information_model_version>
    <product_class></product_class>
    <Alias_List>
    <Citation_Information>
    <Modification_History>
</Identification_Area>
```

The first two attributes, **logical_identifier** and **version_id**, are for the LID and VID values that were discussed in Section 5, Assigning Unique Identifiers. The **title** attribute should be assigned a value that applies specifically to this individual data product. The **information_model_version** must match the version of the PDS4 schema specified in the root tag, e.g. 1.7.0.0 for schema PDS4_PDS_1700. The **product_class** must match the root tag, e.g. **Product_Observational**.

The **Alias_List** class allows you to specify one or more aliases (other identifiers) for this product. **Citation_Information** gives information needed to cite the product in scientific publications. Both classes are optional, and we will omit them in our example.

Modification_History provides a way to track versions of the product after it has been archived. It is also optional, but we'll include it here to demonstrate.

So now our example label includes this **Identification_Area**, using a made-up value for the LID:

```
<Identification_Area>
    <logical_identifier>
       urn:nasa:pds:examples:tables:Table_Character_Example
    </logical_identifier>
    <version_id>1.0</version_id>
    <title>Example of an ASCII fixed-width table in the tables collection</title>
    <information_model_version>1.7.0.0</information_model_version>
    <product_class>Product_Observational</product_class>
    <Modification_History>
        <Modification_Detail>
            <modification_date>2017-03-23</modification_date>
            <version_id>1.0</version_id>
            <description>Initial version</description>
        </Modification_Detail>
    </Modification_History>
</Identification_Area>
```

F.2.5 Modifying Observation_Area

There's a lot going on in this part of the label. **Observation_Area** is where you'll find most of the detailed meta-data about the product. Let's start by opening the `<Observation_Area>` tag but leaving the subclasses closed so that we can get a high-level view.

```
<Observation_Area>
    <comment></comment>
    <Time_Coordinates>
    <Primary_Result_Summary>
    <Investigation_Area>
    <Observing_System>
    <Target_Identification>
    <Mission_Area>
    <Discipline_Area>
</Observation_Area>
```

Remember that this is an exercise in label editing, not a comprehensive guide to the appropriate use of every class and attribute. We'll be skipping most of the optional classes and attributes here, and calling attention to some that you're most likely to need help with. In particular, we'll omit the optional attribute **comment** and the optional class **Primary_Result_Summary**.

F.2.5.1 How To Specify a Nil Value

The **Time_Coordinates** class in the **Observation_Area** is required. It has to include the **start_date_time** and **stop_date_time** attributes, although these can be left blank if they are not available. A value left blank is called a *nil* value. There are two things to remember about using nil values:

1. An attribute can have a nil value only if its definition allows.
2. You have to give a reason why the value is nil.

Checking the definition of **start_date_time** (by right-clicking on the tag in the editor) shows that it is nillable:

```
<xs:element name="start_date_time" nillable="true" type="pds:start_date_time"
minOccurs="1" maxOccurs="1"> </xs:element>
```

When inserting a label attribute with a nil value, use a statement like this to specify the reason:

```
<start_date_time xsi:nil="true" nilReason="inapplicable"></start_date_time>
```

The allowed values for nilReason are:

- inapplicable
- anticipated
- missing
- unknown

Let's assume that the data product in our example has a start time, but typically stop times are not recorded for these observations. So we'll say that **stop_date_time** is nil because it is inapplicable. Our `<Time_Coordinates>` tag now looks like this:

```
<Time_Coordinates>
    <start_date_time>2016-04-19T12:01:33Z</start_date_time>
    <stop_date_time xsi:nil="true" nilReason="inapplicable"></stop_date_time>
</Time_Coordinates>
```

F.2.5.2 Investigation Area and Observing System (or What Context Product LIDs Are For)

The `<Investigation Area>` tag records information about the mission or other coordinated data collection project for which the data product was acquired. The `<Observing_System>` tag records information about the instrument(s) that acquired the data product. These are not places to describe the project and instrumentation in detail. They simply identify the project and instruments and point to other resources where more information can be found. The other resources may be external to PDS in the form of published documents, or they may be internal to PDS in the form of instrument context products, as discussed in Section 9.

Here are the `<Investigation_Area>` and `<Observing_System>` tags for our example product. We'll assume the data collection was done by an individual investigator, and a single instrument component was used to acquire the data. The context product LIDs in this example are made up.

```
<Investigation_Area>
    <name>Spectral Properties of Planetary Glasses</name>
    <type>Individual Investigation</type>
    <Internal_Reference>
        <lid_reference>
           urn:nasa:pds:joe_investigator_spectra:document:about_this_archive
        </lid_reference>
        <reference_type>data_to_investigation</reference_type>
```

```
        </Internal_Reference>
    </Investigation_Area>
    <Observing_System>
        <name>RELAB</name>
        <description>Brown University Reflectance Spectroscopy Laboratory
        </description>
        <Observing_System_Component>
            <name>RELAB Bidirectional Reflectance Spectrometer</name>
            <type>Instrument</type>
            <Internal_Reference>
                <lid_reference>urn:nasa:pds:instrument:bdrs.relab</lid_reference>
                <reference_type>is_instrument</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
    </Observing_System>
```

F.2.5.3 Target_Identification

The `<Target_Identification>` tag is required. It gives the name and type of the target of the observation, along with an optional description and internal reference. The target can be a planetary body such as a planet, satellite, or asteroid, or a laboratory sample, or many other things. See the complete list of target types in the *Data Dictionary* [3] by searching for "type in Target_Identification". In our example the target is a synthetically-generated laboratory sample.

Here is our `<Target_Identification>` tag.

```
    <Target_Identification>
        <name>ALK-2</name>
        <type>Synthetic Sample</type>
    </Target_Identification>
```

We have omitted the `<Internal_Reference>` tag because this laboratory sample does not have a PDS context product that we can refer to. If the target of the observation had been a planet, say Mars for example, then the `<Target_Identification>` tag would include the optional `<Internal_Reference>` tag with the LID for the Mars context product, like this.

```
    <Target_Identification>
      <name>Mars</name>
      <type>Planet</type>
      <Internal_Reference>
        <lid_reference>urn:nasa:pds:context:target:planet.mars</lid_reference>
        <reference_type>data_to_target</reference_type>
      </Internal_Reference>
    </Target_Identification>
```

F.2.5.4 Mission_Area and Discipline_Area are for Local Data Dictionaries

Local data dictionaries were introduced in Section 6.3. Depending on your data products, you may not need to use any of the existing Local Data Dictionaries, much less create a new one, but if you do, this is the place in the label where those dictionary elements – classes and attributes – are placed. In particular, elements from a mission data dictionary go into the Mission_Area and elements from any other Local Data Dictionaries go into the Discipline_Area.

The creation and use of Local Data Dictionaries is a broad topic that is covered in Appendix K of this handbook. For the purpose of this label-editing demonstration we will omit them from our example. This completes the modification of the Observation_Area part of the label.

F.2.6 Modifying Reference_List

The final top-level tag to modify in our example is <Reference_List>. It is optional, but we'll include it for demonstration. Here is how it looks fully expanded.

```
    <Reference_List>
        <Internal_Reference>
            <lid_reference></lid_reference>
<!--            <lidvid_reference></lidvid_reference>-->
            <reference_type></reference_type>
            <comment></comment>
        </Internal_Reference>
        <External_Reference>
            <doi></doi>
            <reference_text></reference_text>
            <description></description>
        </External_Reference>
    </Reference_List>
```

The <Reference_List> tag can contain any number of <Internal_Reference> and/or <External_Reference> tags. Internal references are LIDs or LIDVIDs for resources within PDS. (Notice that <lidvid_reference> is enclosed in comment delimiters. Either <lid_reference> or <lidvid_reference> is required, but not both.) External references lead to resources outside of PDS. See Section 8, Documenting the Archive.

We'll include a made-up external reference in our example label.

```
    <Reference_List>
        <External_Reference>
            <doi>doi:10.1002/2016JE987654321</doi>
            <reference_text>Investigator, J.O., 2017, Spectral Properties
                of Laboratory-Synthesized Glasses, Journal of Geophysical
                Research - Planets, doi:10.1002/2016JE987654321</reference_text>
        </External_Reference>
    </Reference_List>
```

This completes the body of our example label. Both the original draft and the completed version can be found in the *DPH Example Products* at https://pds.nasa.gov/pds4/doc/examples/. The next section shows how to modify the XML prolog and the root tag, which are important for validating the label.

F.3 Editing the XML Prolog and Root Tag

This section addresses how the XML prolog and root tag of an XML label are formed. Both are always found at the beginning of a PDS4 XML label. Here again is our example label with all tags below the root tag closed up.

```
<?xml version="1.0" encoding="UTF-8"?>
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
```

```
    <Observation_Area>
    <Reference_List>
    <File_Area_Observational>
    <File_Area_Observational_Supplemental>
</Product_Observational>
```

F.3.1 XML Declaration Statement

The first XML tag in our label declares the version and encoding of the XML document; it is required, and is created automatically from the PDS4 common schema (PDS4_PDS_nnnn.xsd), if the label has been generated by an XML-aware editor. Typically, it looks like this:

```
<?xml version="1.0" encoding="UTF-8">
```

F.3.2 Schematron References

Now we have to add an XML tag manually to tell the label-reading software how to locate the PDS4 common Schematron file. Enter this just after the first XML tag.

```
<?xml-model href="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

The `href` XML attribute points to the PDS4 common Schematron file (the **.sch** file). Use the latest released version of the Schematron file unless you have a particular reason not to.

If the label will use any Local Data Dictionaries (LDDs), then their associated Schematron files must be referenced using similar notation. These XML tags must be added immediately after the common Schematron XML tag. Our example label for a reflectance spectrum does not use any LDDs, but suppose we were labeling an image data product and we needed to use classes and attributes from the PDS Imaging Node Local Data Dictionary. In that case we would add a third XML tag to specify the Schematron file for that LDD:

```
<?xml-model href="https://pds.nasa.gov/pds4/img/v1/PDS4_IMG_1100.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

All currently available PDS4 schemas and Schematron files can be found online at https://pds.nasa.gov/pds4/schema/released/.

Another resource for information on the XML Prolog is http://sbndev.astro.umd.edu/wiki/Anatomy_of_the_XML_Prolog at the PDS Small Bodies Node.

F.3.3 Local File References and Catalog Files

Both of the above Schematron references point to an online location where the Schematron file is found. It is possible to point to a file on your local system instead. Reasons why you might want to do that include:

1. You are working offline.
2. You want to use a development version of a Schematron file that is available locally but not yet online.
3. Your XML-aware editor does not recognize an online reference.[13]

[13] The oXygen editor does recognize online references. Some versions of the Eclipse editor do not.

In the end, when you deliver your labels to PDS they must use online references to both schemas and Schematron files. It would be cumbersome to edit every label each time you switched back and forth from local to online access during your label development. To get around this, an XML *catalog file* may be used to map the permanent location listed in the label to a working location on your file system. A catalog file is used only by an XML-aware editor when opening an XML file; the catalog file is not part of the PDS archive. Both oXygen and Eclipse editors support XML catalog files. See Appendix I for instructions on setting up a catalog file.

F.3.4 Product Type and Namespaces in the Root Tag

The next statement in the label is the root tag for **Product_Observational**.

```
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
```

This statement was generated automatically by oXygen when it was created using the procedure in Section E.3. In step 2.e, **Product_Observational** was selected as the root element.

Inside this `<Product_Observational>` tag are two namespace references, which begin with `xmlns`, and one schema location reference, which begins with `xsi:schemaLocation`.

The reference `xmlns="http://pds.nasa.gov/pds4/pds/v1"` specifies the default namespace for the label, the PDS namespace. All PDS4 labels should have the default namespace set in this way.

If we needed to include any Local Data Dictionary components in our label, we would add a reference for each LDD namespace after the PDS namespace. For example, if we were using the Imaging Node Local Data Dictionary, we would add the **img** namespace, and the root tag would then look like this:

```
<Product_Observational xmlns=https://pds.nasa.gov/pds4/pds/v1
xmlns:img="http://pds.nasa.gov/pds4/img/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd">
```

The next namespace reference, `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`, specifies the XML Schema Instance namespace **xsi**, which helps the label reading software interpret the schemas used in the label. This is a bit of boilerplate that should be present in every PDS4 XML label.

F.3.5 Schema Locations in the Root Tag

The reference

```
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
file:/Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd"
```

tells label-reading software where to look for the PDS4 common schema. The value of `<xsi:schemaLocation>` is always given as one or more pairs of character strings. The first string in the pair is a namespace – in this case, the PDS4 common namespace. The second string is the location of the schema file (the **.xsd** file) for that namespace. Notice that our label was

created with a local file reference for the schema file (starting with `file:`). We could leave it that way for now, but let's change the local reference to the permanent online location of the PDS4 schema file. (The discussion in Section F.3.3 above about using an XML catalog file to tell your editor where to look for Schematron files applies to schema files as well.) Now we have

```
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd"
```

The `xsi:schemaLocation` value may have several namespace-schema pairs, one pair for the PDS4 common schema and one pair for each Local Data Dictionary used in the label. If our label needed to use the Image Node Local Data Dictionary, we would include the **img** namespace and its location, like this:

```
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                    https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd
                    http:/pds.nasa.gov/pds4/img/v1
                    https://pds.nasa.gov/pds4/pds/v1/PDS4_IMG_1100.xsd"
```

Notice there is no punctuation between namespace names and schema file names, only white space. The white space can be arranged to make the pairs easier to read, as in the above example.

F.3.5.1 Use http or https?

You may have noticed the use of both `http` and `https` in the above examples. It's confusing because the actual *name* of a namespace looks very much like the URI of a file location. In the above example, `http://pds.nasa.gov/pds4/pds/v1` is the name of the PDS4 common namespace, and `https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd` is the location of the schema file for the namespace, an actual URI. Namespace names in PDS generally start with `http`. File location URIs may start with either `http` or `https` depending on the server providing the file. File locations under `pds.nasa.gov` always start with `https`.

F.3.6 The End Tag

Finally, don't forget to end the `<Product_Observational>` tag with its right angle bracket after the final schema location value, and end the entire label with the matching closing tag `</Product_Observational>`. Our example label now looks like this, with all but the root tag closed up.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                    https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Observation_Area>
    <Reference_List>
    <File_Area_Observational>
    <File_Area_Observational_Supplemental>
</Product_Observational>
```

Both the original draft of the example label (**Table_Character_draft1.xml**) and the completed version (**Table_Character_Example**) can be found in the *DPH Examples* at https://pds.nasa.gov/pds4/doc/examples/.

F.4 Turning a Label into a Label Template

Now that our example reflectance spectrum label is complete and valid based on the PDS4 schema and Schematron, we will use it as the basis for a label template.

Going through the label we can identify the attributes that we expect to have different values from one reflectance spectrum data file to the next. They are:

- `<logical_identifier>`
- `<title>`
- `<modification_date>`
- `<start_date_time>`
- `<name>` in `<Target_Identification>`
- `<type>` in `<Target_Identification>`
- `<file_name>` in `<File_Area_Observational>`
- `<file_name>` in `<File_Area_Observational_Supplemental>`
- `<creation_date_time>`
- `<records>` in `<File>`
- `<records>` in `<Table_Character>`

For each of these we will replace the value currently in the label with a placeholder value that has the name of the field delimited by character strings that software will recognize. For example, we'll replace

```
<start_date_time>2016-04-19T12:01:33Z</start_date_time>
```

with

```
<start_date_time>${start_date_time}</start_date_time>.
```

A note on placeholders: the way you represent a placeholder in your label template will depend on the requirements of the software that is going to read it. We chose the delimiters `${` and `}`, but there is no single correct way to do it. If you're planning to use an existing label generation tool, find out its requirements for placeholders. If you're writing your own label generation software you're free to choose a placeholder syntax. Be aware that inserting placeholders instead of actual values may cause your label template to appear invalid in the XML-aware editor. For instance, when we substitute the string `${start_date_time}` in our example label, the editor will show this as a validation error because "${start_date_time}" is not a valid date-time value.

The completed label template can be found in the *DPH Examples* as **Table_Character_Example_template.xml**.

Appendix G Using Local_Internal_Reference and local_identifier

The **Local_Internal_Reference** class and **local_identifier** attribute allow one part of a label to refer to an object described in another part of the same label. (This is not to be confused with the **Internal_Reference** class discussed in Section 8.3.) Many **Product_Observational** subclasses include the optional attribute **local_identifier**, which may be set to any name you choose, with the intent of using that name in another part of the label that needs to refer to the product. (The term "local" here means local within the label.)

A common use of **Local_Internal_Reference** and **local_identifier** is in the Display Local Data Dictionary. For example, if you have an image product described as an **Array_2D_Image**, you need to indicate how the image should be displayed. The Display LDD includes the **Display_Settings** class for this purpose. The relevant parts of the image label might look like the excerpt below.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
<?xml-model href="https://pds.nasa.gov/pds4/disp/v1/PDS4_DISP_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>

<Product_Observational xmlns="http://pds.nasa.gov/pds4/pds/v1"
    xmlns:disp="http://pds.nasa.gov/pds4/disp/v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
       http://pds.nasa.gov/pds4/pds/v1
       https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd
       http://pds.nasa.gov/pds4/disp/v1
       https://pds.nasa.gov/pds4/disp/v1/PDS4_DISP_1700.xsd
...
    <Observation_Area>
...
        <Discipline_Area>
            <disp:Display_Settings>
                <Local_Internal_Reference>
                    <local_identifier_reference>
                        Image XYZ</local_identifier_reference>
                    <local_reference_type>
                        display_settings_to_array</local_reference_type>
                </Local_Internal_Reference>
                <disp:Display_Direction>
                    <disp:horizontal_display_axis>
                        Sample</disp:horizontal_display_axis>
                    <disp:horizontal_display_direction>
                        Left to Right</disp:horizontal_display_direction>
                    <disp:vertical_display_axis>
                        Line</disp:vertical_display_axis>
                    <disp:vertical_display_direction>
                        Top to Bottom</disp:vertical_display_direction>
                </disp:Display_Direction>
            </disp:Display_Settings>
...
        </Discipline_Area>
    </Observation_Area>
    <File_Area_Observational>
...
        <Array_2D_Image>
            <name>Example Image</name>
```

```
            <local_identifier>Image XYZ</local_identifier>
            <offset unit="byte">0</offset>
            <axes>2</axes>
            <axis_index_order>Last Index Fastest</axis_index_order>
            <Element_Array>
                <data_type>IEEE754LSBSingle</data_type>
                <unit>degree</unit>
            </Element_Array>
            <Axis_Array>
                <axis_name>Line</axis_name>
                <elements>3387</elements>
                <sequence_number>1</sequence_number>
            </Axis_Array>
            <Axis_Array>
                <axis_name>Sample</axis_name>
                <elements>3387</elements>
                <sequence_number>2</sequence_number>
            </Axis_Array>
        </Array_2D_Image>
    </File_Area_Observational>
</Product_Observational>
```

The Display LDD (namespace **disp**) is specified in the XML prolog and the `<Product Observational>` root tag, and its classes and attributes are used in the `<Discipline_Area>` tag. The image is described in the `<Array_2D_Image>` tag. Notice that the image's **local_identifier** attribute is set to "Image XYZ", and that this value also appears in the **local_identifier_reference** attribute of the **Local_Internal_Reference** class inside the **disp:Display_Settings** class, as highlighted in red text. In English, this means the display settings of samples left to right, lines top to bottom apply to the image identified in the label as "Image XYZ". This is a simple example. Imagine a product consisting of multiple images that might require different display settings; using local identifiers it is possible to specify different settings for the different images.

Local_Internal_Reference has two required attributes, **local_identifier_reference** and **local_reference_type** (and an optional **comment** attribute). The first is the value of a **local_identifier** that appears somewhere else in the label. The second is the type of reference, in this example "display_settings_to_array". Values for **local_reference_type** should be defined by the Schematron rules for the LDD.

Appendix H Counting Fields and Groups in a Nested Structure

Section F.2.2.5 of Appendix F shows how to repeat a class or attribute in a label, using **Field_Character** in a table as an example. It mentions that tables may have groups of fields that repeat, using the **Group_Field_Character, Group_Field_Delimited,** or **Group_Field_Binary** classes. In these cases it can be confusing to know how to count the byte offsets for fields and groups in order to enter the correct values for the **field_location** and **field_length** attributes. This appendix explains how to do that.

In a table with fixed-width columns (**Table_Character** or **Table_Binary**) it is possible to have a group of fields (i.e., columns) that repeats. It's also possible for a group to contain another group. With a nested structure like this it can be hard to figure out what to put in the label for the attributes **field_location** (starting byte of a field) and **group_location** (starting byte of a group) for each field in the table.

Figure G-1 illustrates how to count the bytes in a label for a table with nested groups. For simplicity, we have assumed that every field has length 4 bytes and that the Group_Field_Z1 begins at **group_location** 0. The bullets below show the number of fields in the various groups; multiplication by 4 gives the number of bytes. Rather than numbers, the content of each XML element is given as an arithmetic expression for **field_location**, **field_length**, **group_location**, and **group_length**.

- Group_Field_Z1 has J fields plus B1 plus B2 repeated L times
  = {J + B1 + B2}*L total fields
  = {J + W*M + [N + (Q + T*V)*U]*S}*L total fields

- Group_Field_B1 has W fields repeated M times
  = W*M total fields

- Group_Field_B2 has N fields plus E1 repeated S times
  = [N + E1]*S total fields
  = [N + (Q + T*V)*U]*S total fields

- Group_Field_E1 has Q fields plus G1 repeated U times
  = (Q + G1)*U total fields
  = (Q + T*V)*U total fields

- Group_Field_G1 has T fields repeated V times

= T*V total fields

Note that the minimum value for <fields> is “0” and the sum <fields> + <groups> must be at least 1.

```
<Group_Field_Z1>
      <group_length>L{4J+4MW+[4N+4(VT+4Q)U]S}</group_length>
      <group_location>0</group_location>
      <fields>J</fields>
      <groups>2</groups>
      <Field_A1>
            <field_length>4</field_length>
            <field_location>0</field_location>
      </Field_A1>
      <Field_A2>
            <field_length>4</field_length>
            <field_location>4</field_location>
      </Field_A2>
                  ...
      <Field_AJ>
            <field_length>4</field_length>
            <field_location>4(J-1)</field_location>
      </Field_AJ>
      <Group_Field_B1>
            <group_length>4MW</group_length>
            <group_location>4J</group_location>
            <fields>W</fields>
            <groups>0</groups>
            <Field_C1>
                  <field_length>4</field_length>
                  <field_location>0</field_location>
            </Field_C1>
            <Field_C2>
                  <field_length>4</field_length>
                  <field_location>4</field_location>
            </Field_C2>
                        ...
            <Field_CW>
                  <field_length>4</field_length>
                  <field_location>4(W-1)</field_location>
            </Field_CW>
            <repetitions>M</repetitions>
      </Group_Field_B1>
      <Group_Field_B2>
            <group_length>[4N+4(VT+4Q)U]S</group_length>
            <group_location>4J+4MW</group_location>
            <fields>N</fields>
            <groups>1</groups>
            <Field_D1>
                  <field_length>4</field_length>
                  <field_location>0</field_location>
            </Field_D1>
            <Field_D2>
                  <field_length>4</field_length>
                  <field_location>4</field_location>
            </Field_D2>
                        ...
```

```
            <Field_DN>
                  <field_length>4</field_length>
                  <field_location>4(N-1)</field_location>
            </Field_DN>

            <Group_Field_E1>

                  <group_length>(4VT+4Q)U</group_length>
                  <group_location>4N</group_location>
                  <fields>Q</fields>
                  <groups>1</groups>

                  <Field_F1>
                        <field_length>4</field_length>
                        <field_location>0</field_location>
                  </Field_F1>

                  <Field_F2>
                        <field_length>4</field_length>
                        <field_location>4</field_location>
                  </Field_F2>
                              ...
                  <Field_FQ>
                        <field_length>4</field_length>
                        <field_location>4(Q-1)</field_location>
                  </Field_FQ>

                  <Group_Field_G1>

                        <group_length>4VT</group_length>
                        <group_location>4Q</group_location>
                        <fields>T</fields>
                        <groups>0</groups>

                        <Field_H1>
                              <field_length>4</field_length>
                              <field_location>0</field_location>
                        </Field_H1>

                        <Field_H2>
                              <field_length>4</field_length>
                              <field_location>4</field_location>
                        </Field_H2>
                                    ...
                        <Field_HT>

                              <field_length>4</field_length>
                              <field_location>4(T-1)</field_location>
                        </Field_HT>

                        <repetitions>V</repetitions>

                  <Group_Field_G1>

                  <repetitions>U</repetitions>

            </Group_Field_E1>

            <repetitions>S</repetitions>

      </Group_Field_B2>

      <repetitions>L</repetitions>

</Group_Field_Z1>
```

Figure H-1. Nested Groups and Fields in a Table Label

Appendix I Using XML Catalog Files to Locate Schema

As discussed in Section F.3, the **xsi:schemaLocation** attribute of the root tag contains pairs of values where the first value is a namespace and the second is the location of the XML schema for that namespace. For example:

```
xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                    https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">
```

The location can be the path and file name of a schema on your local file system, or it can be a URI that leads to the schema on another system. The final, archived version of the label must use the official URIs for the PDS4 common schema and other schemas; no references to local files are permitted. But there are times during label development when you might want to use a local version of a schema; perhaps you are using the schema for a new LDD that is not yet published at PDS, or perhaps you are working offline. An XML catalog file provides a way for you to use a local schema even when your label specifies an official URI. These catalog files are used by some XML-aware editors, including oXygen and Eclipse, to translate the schema location specified in the label to a different location specified in the catalog file.

Here is an example of an XML catalog file that causes the editor to use local copies of the PDS common schema and Schematron files instead of the ones online at PDS.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <uri name="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd"
         uri="file:///Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd"/>
    <uri name="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
         uri="file:///Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.sch"/>
</catalog>
```

When this catalog file has been registered in the XML-aware editor[14], wherever the editor sees the schema location `https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd` in the label, it will substitute the schema location `file:///Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.xsd`. It will do the same for the location of the Schematron file `file:///Z:/stage/pds4/schema/pds/v1/PDS4_PDS_1700.sch`.

Here is an even simpler method.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <rewriteURI uriStartString="https://pds.nasa.gov/pds4"
                rewritePrefix="file:///Z:/stage/pds4/schema"/>
</catalog>
```

[14] In oXygen, select Options -> Preferences -> XML Catalog -> Add, and browse to the location of your catalog file.

With this catalog file, the editor will substitute the string `file:///Z:/stage/pds4/schema` where it finds the string `https://pds.nasa.gov/pds4` in the label.

A catalog file may contain multiple `uri` and/or `rewrite` tags. When the editor is processing the file, it will stop at the first statement that matches, so they should be arranged in order from specific to general.

There are others ways to use catalog files to redirect the editor, but these are two of the most common. See the PDS Small Bodies Node page *Understanding XML Catalog Files* at http://sbndev.astro.umd.edu/wiki/Understanding_XML_Catalog_Files for a fuller explanation.

Once the XML catalog file has been created, it will need to be registered with the XML-aware editor that you are using. (Note that not all XML-aware editors accommodate catalog files, and the ones that do may interact with them differently.) Once registered, you will also need to ensure that the XML catalog file is being referenced by your XML product labels, possibly by re-validating the product label, or closing and re-opening the label, or closing and re-opening the editor.

Appendix J Using Schematron Rules to Help Validate Labels

This appendix introduces the use of Schematron rules to help validate PDS4 labels. Section 11.1, Label Validation, refers to this appendix. Using Schematron rules is a rather advanced skill, so if you are new to this topic, ask for help from your consulting node.

In the field of markup languages, **Schematron** is a rule-based validation language for making assertions about the presence or absence of patterns in XML trees. It is a structural schema language expressed in XML using a small number of XML elements and XPath, which is a syntax for expressing the path to elements in an XML document (https://www.w3.org/TR/xpath-30/).

Schematron is capable of expressing constraints in ways that XML Schema cannot. For example, it can require that the content of an XML element be controlled by one of its siblings. Or it can require that the root element, regardless of what element that is, have specific attributes.

Constraints and content rules may be associated with plain English validation error messages, allowing translation of numeric Schematron error codes into meaningful user error messages.

One possible use of Schematron is to ensure that an optional class or attribute is included in the label. You can do this by adding a rule to a Schematron file. For example, the optional attribute **description** was omitted from the **Field_Character** classes in the label exercise in Appendix F. If you wanted to ensure that **description** was always included in **Field_Character** in your labels, you could create a Schematron rule, such as the one below, that would generate an error message if **description** were missing. In the example, the text in the `<sch:assert>` tag is the error message that would be displayed.

You would need to work with your consulting node to add this constraint to a Local Data Dictionary Schematron file, since you may not modify the PDS4 common Schematron file.

```
<sch:pattern>
      <sch:rule context="//pds:Field_Character">
            <sch:assert test="pds:description">
                  The description in Field_Character must exist.
            </sch:assert>
      </sch:rule>
</sch:pattern>
```

The following are good sources of additional information:

- Roger Costello has posted a good set of introductory tutorials. There is no need to run through all of the tutorials. But "Two Types of XML Validation", "Usage and Features", and "Overview" will set the stage for what Schematron is.
  - http://www.xfront.com/schematron/index.html
- Miloslav Nic has posted a great set of starter examples and how-to information at:
  - http://zvon.org/xxl/SchematronTutorial/General/toc.html
- You can download and walk through "A hands-on introduction to Schematron" by Uche Ogbuji from:
  - https://www6.software.ibm.com/developerworks/education/x-schematron/x-schematron-a4.pdf

Appendix K Creating and Using Local Data Dictionaries

This appendix has not been updated for PDS4 1.7.0.0 and its instructions may not represent the current best practice. For help creating and using Local Data Dictionaries, ask your PDS node representative.

This appendix gives instructions on creating and using Local Data Dictionaries. See Section 6.3 for an overview of data dictionaries in PDS4.

PDS provides a tool for generating Local Data Dictionary schemas called LDDTool, available from the PDS4 Software page of the PDS web site (https://pds.nasa.gov/pds4/software/ldd/). It's possible to create a Local Data Dictionary schema without using LDDTool, but using the tool ensures compatibility with the PDS4 common schema. The LDDTool software package includes the *Ingest LDD Users Guide* [8].

A local dictionary must reside within a namespace that is unique across all other locally defined dictionaries. It is critical to avoid collisions among namespaces either being used or reserved for use in PDS4 labels. Refer to the *Standards Reference* [2] Section 6B and confer with your PDS consulting node to determine an appropriate namespace if you create a local dictionary. The currently registered namespaces can be found at:

https://pds.nasa.gov/pds4/schema/pds-namespace-registry.pdf

Note that this list is under development; some of the namespaces shown do not conform to the naming constraints in *Standards Reference* Section 6B.2.

For additional information regarding the process for either adding or reserving a new namespace, consult your discipline node or contact the PDS Engineering Node.

K.1 The Mission Area

The Mission_Area, an optional class within the Observation_Area, functionally acts as a container for mission specific classes and attributes, each of which is defined in a Local Data Dictionary. The set of locally defined attributes and classes must be prefixed with a unique mission namespace identifier applicable to the respective dictionary. In the example below `spacecraft_clock_start_count` and `spacecraft_clock_stop_count` have been defined in the dph namespace:

```
<Observation_Area>
      .
      .
      .
 <Mission_Area>
    <dph:spacecraft_clock_start_count>
       1246943630
    </dph:spacecraft_clock_start_count>
    <dph:spacecraft_clock_stop_count>
       1246943631
```

```
      </dph:spacecraft_clock_stop_count>
    </Mission_Area>
    <Discipline_Area>
    </Discipline_Area>
  </Observation_Area>
```

K.2 The Discipline Area

The Discipline_Area, another optional class within the Observation_Area, functionally acts as a container for discipline specific classes and attributes, each of which is defined in a Local Data Dictionary. The set of locally defined attributes and classes must be prefixed with a unique discipline namespace identifier applicable to the respective dictionary. In the example below `application_process_id` and `application_process_name` are defined in the img namespace:

```
  <Observation_Area>
        .
        .
        .
    <Mission_Area>
        .
        .
        .
    </Mission_Area>
    <Discipline_Area>
      <img:application_process_id>
        AGP17
      </img:application_process_id>
      <img:application_process_name>
        Non-linear stretch algorithm 17
      </img:application_process_name>
    </Discipline_Area>
  </Observation_Area>
```

K.3 Merging Local Data Dictionaries With the Common Dictionary

PDS can merge the attributes and classes defined in a Local Data Dictionary with the PDS4 Information Model. Merging ensures that your locally defined attributes and/or classes will be contained in future builds of the PDS4 Data Dictionary and generic mission and discipline schemas. Merging is optional but, once completed, means that your locally defined classes and attributes are readily available any time you need them. A successful merge requires, however, that the local definitions be compatible with the common structure and contents. PDS provides LDDTool to facilitate this process.

The input to LDDTool is an XML file based on the Ingest_LDD class in the common schema. As producer of the local dictionary, you generate and populate the input file, run LDDTool, and submit both the input file and the output files to PDS. Details of this process are contained in the *Ingest LDD Users Guide*, which is included in the LDDTool package (https://pds.nasa.gov/pds4/software/ldd/). We summarize key steps below.

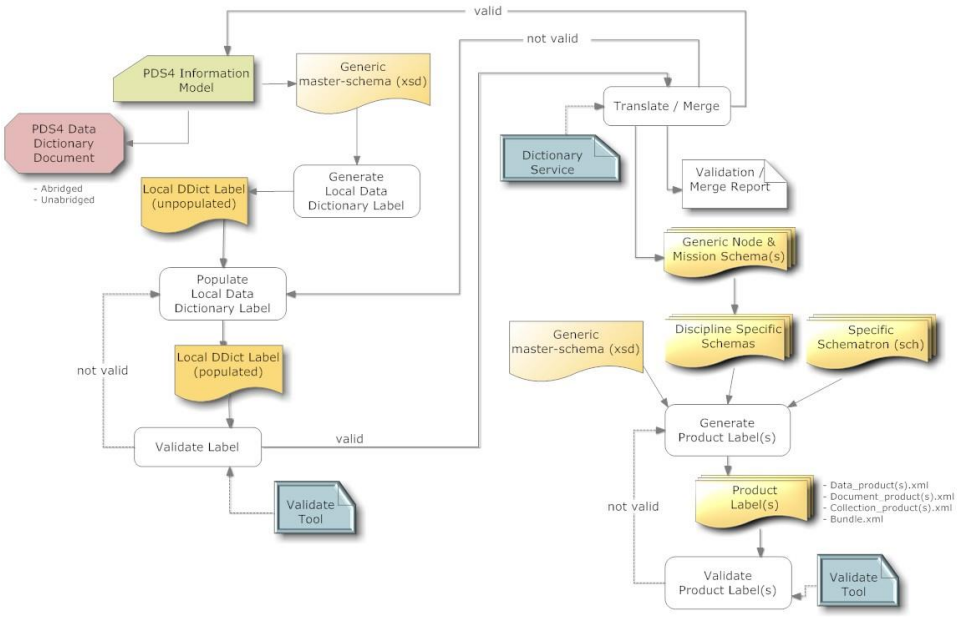

Figure I-1. Local data dictionary development and label production flow diagram.

Step #1: Download the current version of the common-schema to a directory/folder on your local computer. Files will have names of the form PDS4_PDS_nnnn.xsd and can be found at:

https://pds.nasa.gov/pds4/schema/released/pds/

Step #2: Using the XML-aware editor (XAE) of your choice, open the common-schema (XSD) file. Locate the Ingest_LDD class. This is the class from which we will generate the input XML file which the dictionary tool will use to generate the schema for your dictionary. The process for producing this input file is the same as the process for producing a label template file (see Appendix Appendix F of this document). In this case instead of using Product_* in your Root Tag, you use Ingest_LDD.

Step #3: Most XML-aware editors provide a capability to generate an XML-template from a schema.

In oXygen:

a. select: Tools/Generate Sample XML Files.

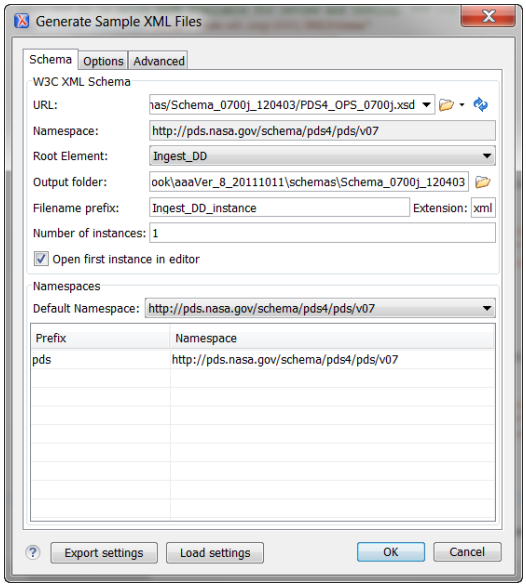

Figure K-1. Using oXygen to Generate a Label Template

You should see a form like that shown in Figure 11-2; notice that the Schema tab is active. This form is used to specify the XSD from which the template file will be generated and to specify the output template file name and location. Fill in the boxes as follows:

a. In "URL" use the folder icon to select the local copy of the common-schema file.

b. In "Namespace" the value should have been auto-populated with the value specified in the common-schema. The value should be fairly close to the value displayed above.

c. In "Root Element" use the drop-down menu to select the product for which you want to create a template file – Ingest_LDD.

d. In "Output folder" specify the directory/folder where you want the template file to reside (when created).

e. In "Filename prefix" specify the name of the template file.

f. In "Number of instances" enter "1". If you enter a larger number, additional copies of the template will be created, each having a name beginning with the value specified in "Filename prefix".

g. Check the "Open first instance in editor" box so that the template file will be displayed in oXygen when created.

h. "Default Namespace" should have been auto-populated with the value specified in the common-schema. The value should be fairly close to the value displayed above.

i. The “Prefix” and “Namespace” areas should have been auto-populated with values fairly close to those displayed above.

Once you have verified the above values, select the Options tab (at the top of the dialog box) and fill in the boxes on the next form:

j. Check the “Generate optional elements” box so all optional elements are included in the template file.

k. Check the “Generate optional attributes” box so all optional attributes are included in the tenplate file.

l. In “Values of elements and attributes” select “Default” using the pull-down menu.

m. In “Preferred number of repetitions” specify “1” so that a single instance of each class and attribute will be included in the template file.

n. In “Maximum recursivity level” specify “1” as the maximum depth in case of recursivity.

o. In “Choice strategy” select “Random” using the pull-down menu.

Once you have verified the above values, click the OK button. This will generate the XML template file, which should be displayed in your XAE window (Figure K-2).

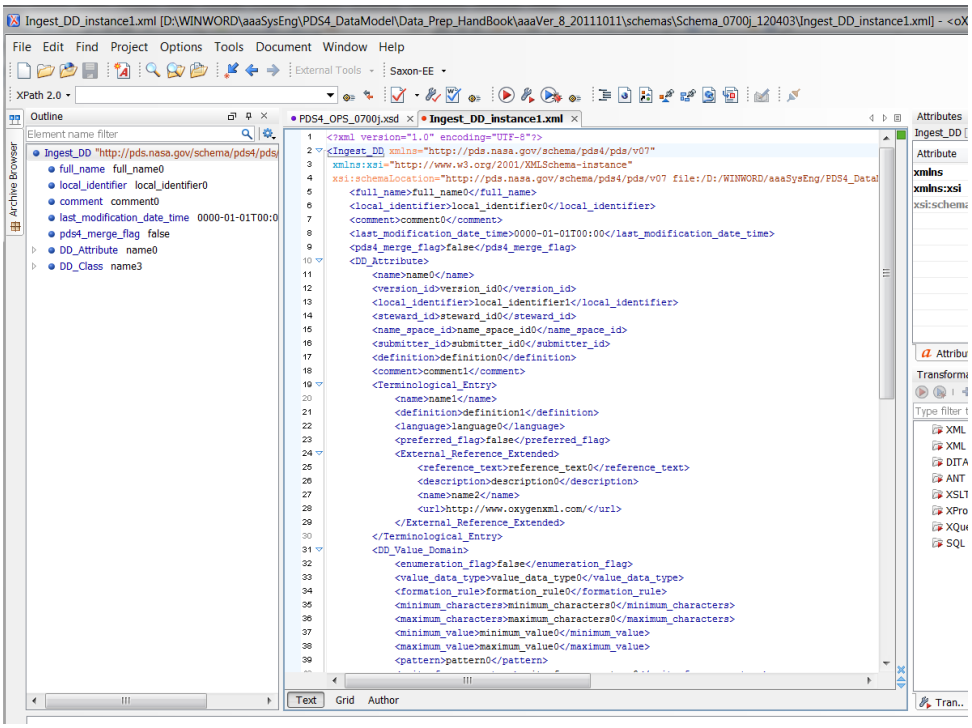

Figure K-2. Template input file for LDDTool as generated by oXygen

If you are using the oXygen XAE, ensure that you can see a little green box in the upper right. If the box is not green, the template is not valid; contact your consulting DN for suggestions on resolving discrepancies.

The template, once populated with real values, will be used by LDDTool to generate your local dictionary product (schema, Schematron, and XML label files).

Step #4: With the template displayed in your XAE, you are now ready to begin populating it with the metadata associated with each attribute and class that is to be defined in the Local Data Dictionary.

As you populate the template, ensure that the XML remains fully formed — that is, keep the box in the upper right green. If the color changes, you have introduced an error in the XML. oXygen will do this validation automatically as you work; other XAEs may have similar real-time validations (or not). You can also use the PDS4 Validation Tool to validate the template.

Your unedited template should look something like the following. It has a standard XML Declaration, a Root Tag, some dictionary overview attributes, and four classes defining the types of content that can be included in the dictionary: DD_Attribute, DD_Class, DD_Rule, and Property_Maps.

```
<?xml version="1.0" encoding="UTF-8"?>
<Ingest_LDD
  xmlns="http://pds.nasa.gov/pds4/pds/v1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1 file:/TEST/PDS4_PDS_1700.xsd">
    <name>name0</name>
    <ldd_version_id>ldd_version_id0</ldd_version_id>
    <full_name>full_name0</full_name>
    <steward_id>steward_id0</steward_id>
    <namespace_id>namespace_id0</namespace_id>
    <external_property_maps_id>
       external_property_maps_id0
    </external_property_maps_id>
    <comment>comment0</comment>
    <last_modification_date_time>0000</last_modification_date_time>

    <DD_Attribute>
    ...
    </DD_Attribute>

    <DD_Class>
    ...
    </DD_Class>

    <DD_Rule>
    ...
    </DD_Rule>

    <Property_Maps>
    ...
    </Property_Maps>

</Ingest_LDD>
```

Step #5: As necessary, edit the XML Declaration and Root Tag. An XML tag is a character string delimited by a pair of angle brackets — "<" and ">" (Section 7.2 of this document).

a. The XML Declaration (line 1 in the example above) needs no editing.

b. The Root Tag correctly references the PDS common dictionary and the XML Schema Instance (xsi) namespace, so lines 2-4 need no editing.

c. If your local dictionary will reference any other discipline dictionary, you need to add that namespace identfication to the Root Tag. For purposes of this example, we will assume you will reference the geometry discipline dictionary. The namespace ID for the geometry dictionary is "geom".

d. You need to add a reference to the Schematron file for each of the dictionaries you plan to reference, including the common dictionary. Do this by inserting one XML tag for each dictionary between the XML Declaration and the Root Tag. As discussed in Section F.3.2, XAEs have different requirements regarding Schematron references; check your XAE user guide or contact your consulting DN if there are problems.

After these changes, your XML Prolog and Root Tag should look like this, where the new lines are shown in red:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"?>
<Ingest_LDD
  xmlns="http://pds.nasa.gov/pds4/pds/v1"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:geom=http://pds.nasa.gov/pds4/geom/v1
  xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                      file:/TEST/PDS4_PDS_1700.xsd">
```

e. The next edit is for the line beginning "xsi:schemaLocation". In the unedited template, the entry refers to a local file. As discussed in Section F.3, the final version should not include local references, although some XAEs make this difficult. Here we assume the use of a catalog file to enable the XAE to reference a local version, while the edited template provides the permanent URI for the schema. The unedited version refers to the common schema; in this example we also need to refer to the geometry schema. The final XML Prolog and Root Tag is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"?>
<?xml-model href="http://pds.nasa.gov/pds4/geom/v1/PDS4_GEOM_1400.sch"?>
<Ingest_LDD
  xmlns="http://pds.nasa.gov/pds4/pds/v1"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:geom=http://pds.nasa.gov/pds4/geom/v1
  xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
                      http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700xsd
                      http://pds.nasa.gov/pds4/geom/v1
                      http://pds.nasa.gov/pds4/geom/v1/PDS4_GEOM_1400.xsd">
```

Step #6: Substitute real values for the placeholders in lines beginning <name> through <last_modification_date_time>.

<name> is the name of your local dictionary (we will use "Phoenix Mission Dictionary" here).

<ldd_version_id> is the version ID of your dictionary. See the *Standards Reference* [2], Section 6D.3.2 for a complete discussion of dictionary and namespace version IDs. The first official delivery to the PDS should have a "1" in the first position.

<full_name> is the full name of the person responsible for developing and maintaining the dictionary.

<steward_id> is the identifier for the agency that is the steward for the dictionary. Discuss this with your consulting node.

<namespace_id> the namespace ID for your dictionary. You may propose an ID to your consulting node; but, in order to ensure no conflicts with existing IDs, the final determination is made by the consulting node and the Engineering Node.

<external_property_maps_id> is optional. Contact your consulting node for more information.

<comment> We strongly recommend you use this attribute to describe your initial release and to document changes in subsequent versions.

<last_modification_date_time> The approximate date and time of the most recent changes.

Step #7: The unedited template contains one copy of the following DD_Attribute class definition.

```
<DD_Attribute>
    <name>name1</name>
    <version_id>0.0</version_id>
    <local_identifier>local_identifier0</local_identifier>
    <nillable_flag>false</nillable_flag>
    <submitter_name>submitter_name0</submitter_name>
    <definition>definition0</definition>
    <comment>comment1</comment>
    <Internal_Reference>
        <lidvid_reference>lidvid_reference111</lidvid_reference>
        <reference_type>reference_type0</reference_type>
        <comment>comment2</comment>
    </Internal_Reference>
    <Terminological_Entry>
        <name>name2</name>
        <definition>definition1</definition>
        <language>language0</language>
        <preferred_flag>false</preferred_flag>
        <instance_id>instance_id0</instance_id>
        <External_Reference_Extended>
            <doi>doi0</doi>
            <reference_text>reference_text0</reference_text>
            <description>description0</description>
            <name>name3</name>
            <url>http://www.oxygenxml.com/</url>
        </External_Reference_Extended>
    </Terminological_Entry>
    <DD_Value_Domain>
        <enumeration_flag>false</enumeration_flag>
        <value_data_type>value_data_type0</value_data_type>
        <formation_rule>formation_rule0</formation_rule>
        <minimum_characters>minimum_characters0</minimum_characters>
        <maximum_characters>maximum_characters0</maximum_characters>
        <minimum_value>minimum_value0</minimum_value>
        <maximum_value>maximum_value0</maximum_value>
```

```
            <pattern>pattern0</pattern>
            <unit_of_measure_type>
               unit_of_measure_type0
            </unit_of_measure_type>
            <specified_unit_id>specified_unit_id0</specified_unit_id>
            <DD_Permissible_Value>
                <value>value0</value>
                <value_meaning>value_meaning0</value_meaning>
                <Terminological_Entry>
                    <name>name4</name>
                    <definition>definition2</definition>
                    <language>language1</language>
                    <preferred_flag>false</preferred_flag>
                    <instance_id>instance_id1</instance_id>
                    <External_Reference_Extended>
                        <reference_text>reference_text1</reference_text>
                    </External_Reference_Extended>
                </Terminological_Entry>
            </DD_Permissible_Value>
        </DD_Value_Domain>
    </DD_Attribute>
```

You will need a complete copy for each attribute you include in your dictionary. However, before you do several dozen copy-and-paste operations, look closely at the contents. Several of the attributes in this class will have the same value in all definitions. Edit these first, and then do the copy and paste. Candidate attributes for copying include version_id (since they all probably start with "1.0") and submitter_name, which is probably the same for most (if not all) of the attributes – usually the individual doing the typing.

All of the attributes in the dictionary must be defined before any of the classes in the dictionary can be defined. For specific information on how to populate each DD_Attribute class, see the *Ingest LDD Users Guide* and the SBN PDS4 Wiki (http://sbndev.astro.umd.edu/wiki/SBN_PDS4_Wiki). Do not hesitate to contact your consulting node.

Step #8: The template sections for DD_Class, DD_Rule, and Property_Map classes can be replicated and edited once you have the DD_Attribute definitions completed. Of course, the process of developing classes and attributes is iterative; you may discover that you need another attribute while you are fleshing out the classes. But you should not end up with attributes that haven't been used in any class and you should never have a class with an attribute that has not been defined. By referencing the PDS common dictionary and selected discipline dictionaries, you can import both attributes and classes (and their definitions) into your own dictionary.

Replicate and edit the template sections for DD_Class, DD_Rule, and Property_Map classes as you did for DD_Attribute; identify values that will not change across your dictionary, edit those parts of the template, and then do your copy-and-paste operation before buckling down to the detail of populating the individual definitions. You can find details in the *Ingest LDD Users Guide*, and the SBN PDS4 Wiki. Do not hesitate to contact your consulting node.

Step #9: At this point, you should have a fully formed (populated) and compliant XML data dictionary input file. Run LDDTool. See the *Ingest LDD Users Guide* for instructions on how to run the tool and what command line flags to set.

Step #10: Review the output files — especially log files — to identify and correct errors. Obtain assistance from your consulting node as necessary. Then repeat Step #9 until you have an error-free run.

Step #11: Once you have an error-free LDDTool run, you should send the input XML file and the output files to the PDS Engineering Node. The PDS EN will use software to ingest the classes and attributes that you defined (in Ingest_LDD.xml) into the IM. EN will then send back to you the following:

a. A local-dictionary XSD that contains the classes and attributes that you defined.
b. A validation report that indicates either success or issues that may still need to be resolved.
c. A local-dictionary HTML file that can be browsed for a human-readable dictionary content.

The successful result is that you will have online access to your local dictionary schema (XSD) at:

https://pds.nasa.gov/pds4/schema/released/.

Appendix L Forming Logical Identifiers (LIDs) for Context Products

This appendix gives rules for forming unique Logical Identifiers (LIDs) for new PDS context products. See Section 5 for a discussion of LIDs and Section 9 for information about context products. Like all PDS products, context products must be assigned identifiers that are unique across PDS. If the data products you are providing are related to instruments, spacecraft, missions, etc., that do not already have PDS context products, then you will need to provide them. Your consulting discipline node will help determine what context products are needed.

A context product LID has the form **urn:nasa:pds:context:[1]:[2].[3]**. This follows the form for regular LIDs, bundle:collection:product. The bundle identifier is **context**. Field [1] is the collection identifier, and fields [2] and [3] form the product identifier.

Use the following table to choose values for fields [1], [2], and [3] for each context product you will create. To guarantee that the LIDs are unique, you or your consulting node should send them to the PDS Engineering Node data engineer who is responsible for approving new LIDs. The data engineer will either approve your choices or suggest better ones.

Notes:

Text in *italics* indicates an actual value to be used exactly as shown.

The delimiter between fields [2] and [3] is a period, not a colon.

Examples are shown with spaces between fields for clarity. Actual LIDs do not have any spaces.

Table L-1. LIDs for Context Products

urn : nasa : pds : context : [1] *:* [2] . [3]

Type of Context Product	**Field [1]**	**Field [2]**	**Field [3]**
Investigation	*investigation*	Type of investigation, one of: *mission* *campaign* *individual* *other*	Investigation name
Example:	*urn : nasa : pds : context :* ***investigation : mission . osiris-rex*** (OSIRIS-REx asteroid sample return mission)		
Instrument Host (spacecraft)	*instrument_host*	Spacecraft type, one of: *lander* *rover* *spacecraft*	Spacecraft name
Example:	*urn : nasa : pds : context :* ***instrument_host : spacecraft . osiris-rex*** (OSIRIS-REx spacecraft)		
Facility (not spacecraft)	*facility*	Facility type, one of: *laboratory* *observatory*	Facility name
Example:	*urn : nasa : pds : context :* ***facility : observatory . mcdonald*** (McDonald Observatory)		

Type of Context Product	Field [1]	Field [2]	Field [3]
Airborne observing system	*airborne*	Airborne observing system type, one of: *aircraft* *balloon* *suborbital*	Observing system name
Example:	*urn : nasa : pds : context :* ***airborne : balloon . bopps*** (Balloon Observation Platform for Planetary Science)		
Telescope	*telescope*	Host of telescope	Name of telescope
Example:	*urn : nasa : pds : context :* ***telescope : bopps . sto*** (Stratospheric Terahertz Observatory telescope on BOPPS)		
Instrument on a spacecraft	*instrument*	Instrument ID	Spacecraft name
Example:	*urn : nasa : pds : context :* ***instrument : navcam . osiris-rex*** (Navcam instrument on OSIRIS-REx spacecraft)		
Instrument not on a spacecraft	*instrument*	A two-part field: instrument provider type, followed by a period, followed by instrument ID. Choose one of: *facility*.instrumentID *investigation*.instrumentID *person*.instrumentID *telescope*.instrumentID	Name of facility, investigation, person, or telescope
Example:	*urn : nasa : pds : context :* ***instrument : telescope . birc . sto*** (BOPPS InfraRed Camera on the STO telescope)		
Resource	*resource*	*Resource*	Name of resource
Example:	*urn : nasa : pds : context :* ***resource : resource . orbital_data_explorer***		

Type of Context Product	**Field [1]**	**Field [2]**	**Field [3]**
PDS Affiliate or PDS Guest	*personnel*	*Personnel*	Person's given name or initial, followed by an underscore, followed by family name
Example:	*urn : nasa : pds : context :* ***personnel : personnel . richard_chen***		

Type of Context Product	Field [1]	Field [2]	Field [3]
Target	*target*	Type of target, one of: *asteroid* *calibration_field* *calibrator* *comet* *dust* *dwarf_planet* *equipment* *galaxy* *lunar_sample* *meteorite* *meteoroid* *meteoroid_stream* *nebula* *planet* *planetary_nebula* *planetary_system* *plasma_cloud* *plasma_stream* *ring* *satellite* *star* *star_cluster* *sun* *synthetic_sample* *terrestrial_sample* *trans-neptunian_object*	name of target
Example:	*urn : nasa : pds : context :* ***target : satellite . amalthea***		

Type of Context Product	Field [1]	Field [2]	Field [3]
Node	*node*	*Node*	Node name, one of: *atmos* *en* *geosciences* *imaging* *naif* *pds-mgt* *ppi-ucla* *psa* *rings* *rs* *sbn*
Example:	*urn: nasa : pds : context :* ***node : node . atmos***		
Agency	*agency*	*Agency*	Agency name, one of: *esa* *jaxa* *nasa* *ros*
Example:	*urn : nasa : pds : context :* ***agency : agency . esa***		

Node and Agency context products are created and maintained by the PDS Engineering Node. Their LIDs are included here for completeness. It is unlikely that any data provider would need to create a new Node or Agency context product LID.

Appendix M Generating Labels for Collection Inventories and Bundles

This appendix gives instructions for creating labels for the two types of aggregate products, **Product_Collection** and **Product_Bundle**, which were introduced in Section 6.2.4.

M.1 Product_Collection

A **Product_Collection** product defines a collection of basic products that are related in some way. It consists of an inventory of its member products and an accompanying label. Observational data, for example, may be gathered into one or more data collections, documents into a document collection, etc.

The mechanics of defining a collection are straightforward; you create an Inventory table, which is a delimited table with two fields. The first field identifies each member as either primary (P) or secondary (S), as defined in the following section. The second field lists the logical identifiers (LIDs or LIDVIDs) of all the products that are members of the collection.

You then create a label that describes the Inventory table. The Inventory label contains a logical identifier (LID) that uniquely identifies the collection It also contains a version identifier (VID) that distinguishes among several versions of the collection product (if there is more than one).

M.1.1 Members of a Collection

The members of a collection are designated as being either primary (P) or secondary (S).

- Every product must be a primary member of one and only one collection.
    - That collection is the one in which the product is first registered in the PDS.
- Products already registered in the PDS may be secondary members of other collections.
    - For example, a collection of Saturn ring images from an imaging experiment may be created as secondary members because the full set of images from that experiment were previously registered in another collection.
- Primary members must be identified in the collection inventory using LIDVIDs.
- Secondary members may be identified in the collection inventory using either LIDs or LIDVIDs based on which is more appropriate for that collection and product (*Standards Reference* [2] Section 9C).

M.1.2 Collection Inventory

As stated, the Inventory table is a two-field **Table_Delimited** object where each row (or record) of the table describes one of the member products of the collection. The first field of each record specifies whether the product is a primary (P) or secondary (S) member of the collection. The second field specifies either the LID or LIDVID of the product. The only permitted delimiter between the two fields is a comma. The Inventory table file name should be of the form **collection[_*].csv** where the optional base name extension can indicate the type of collection — for example, **collection_browse.csv** or **collection_data_raw.csv**.

Here is the Inventory table in the file **collection_eetable_inventory.csv** in the *DPH Examples* (https://pds.nasa.gov/pds4/doc/examples/).

```
P,urn:nasa:pds:izenberg_pdart14_meap:data_eetable:ele_evt_12hr_orbit_2011-2012::1.0
P,urn:nasa:pds:izenberg_pdart14_meap:data_eetable:ele_evt_8hr_orbit_2012-2013::1.0
P,urn:nasa:pds:izenberg_pdart14_meap:data_eetable:ele_evt_8hr_orbit_2014-2015::1.0
```

M.1.3 Generating and Populating a Product_Collection Label

You may use the procedure described in Appendix E to create a draft **Product_Collection** label and modify it using the information in this section. For an example, we will use the label for the Inventory table shown above. It looks like this with the lower-level tags closed up.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
<Product_Collection xmlns="http://pds.nasa.gov/pds4/pds/v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
http://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Context_Area>
    <Collection>
    <File_Area_Inventory>
</Product_Collection>
```

M.1.3.1 Identification Area

The **Identification_Area** class includes the same subclasses that it did in the **Product_Observational** label in Appendix F. The `<logical_identifier>` tag is filled in with the LID for the collection as a whole (Section 5.2). The `<title>` also applies to the collection as a whole. The `<Citation_Information>` is required if the collection is a Mission Science Data Collection (*Standards Reference* [2] Section 9C.3.1). Its **description** attribute should provide a terse (less than 5000 bytes) description of the contents of the collection suitable for display in a web browser.

Here is the **Identification_Area** part of our example **Product_Collection** label.

```
    <Identification_Area>
        <logical_identifier>
            urn:nasa:pds:izenberg_pdart14_meap:data_eetable
        </logical_identifier>
        <version_id>1.0</version_id>
        <title>
           Izenberg PDART 2014 MESSENGER Advanced Products Energetic Electrons
           Table Collection
        </title>
        <information_model_version>1.7.0.0</information_model_version>
        <product_class>Product_Collection</product_class>
        <Citation_Information>
            <author_list>N. Izenberg</author_list>
            <publication_year>2016</publication_year>
```

```
            <keyword>MESSENGER</keyword>
            <keyword>Neutron Spectrometer</keyword>
            <description>
               Izenberg PDART 2014 MESSENGER Advanced Products Energetic
               Electrons Table Collection
        </description>
        </Citation_Information>
        <Modification_History>
            <Modification_Detail>
                <modification_date>2016-04-08</modification_date>
                <version_id>1.0</version_id>
                <description>Initial PDS4 version</description>
            </Modification_Detail>
        </Modification_History>
    </Identification_Area>
```

M.1.3.1.1Citation Information

The **Citation_Information** subclass in the **Identification_Area** provides information that can be used when citing the collection in journal articles, abstract services, and other reference contexts. This class contains attributes that are used in searching for PDS4 collections. A typical use of **Citation_Information** looks like this:

```
        <Citation_Information>
            <author_list>J. Caesar</author_list>
            <editor_list>A. Smith</editor_list>
            <publication_year>2014</publication_year>
            <keyword>Astrometry</keyword>
            <keyword>Gamma Ray</keyword>
            <keyword>Magnetosphere</keyword>
            <keyword>Voyager</keyword>
            <keyword>Jupiter</keyword>
            <keyword>ppi-ucla</keyword>
            <description>
                  Collection of browse products in the PPI VG2PLS archive.
            </description>
         </Citation_Information>
```

The **description** and **publication_year** attributes are required; the others are optional but highly recommended because their inclusion improves the PDS4 search process. The **keyword** attribute is especially useful for listing individual terms that may satisfy search criteria.

M.1.3.2 Context Area

The **Product_Collection** label may include a **Context_Area** class, something that is not present in a **Product_Observational** label. **Context_Area** contains attributes that are used in searching for PDS4 products. This is where you provide reference information that describes the collection and links other PDS4 products to the collection. Although it is optional, it is recommended that you include it to help users who are searching for data products like yours.

Context_Area consists of the following optional[15] subclasses:

- **Time_Coordinates** – start and stop times for the collection

[15] Some of these are required for Mission Science Data Collections. See *Standards Reference* [2] 9C.3.1.

- **Primary_Result_Summary** – high-level description of the types of products in the collection
- **Investigation_Area** – information about the mission or data collection effort (similar to **Investigation_Area** in **Product_Observational**)
- **Observing_System** – information about the instrumentation used to acquire the products in the collection (similar to **Observing_System** in **Product_Observational**)
- **Target_Identification** – information about the target(s) of the observations in the collection (similar to **Target_Identification** in **Product_Observational**)
- **Mission_Area** – a place for classes and attributes from a mission Local Data Dictionary
- **Discipline_Area** – a place for classes and attributes from node or discipline Local Data Dictionaries.

Here is the **Context_Area** part of our example Inventory label.

```
<Context_Area>
    <Time_Coordinates>
        <start_date_time>2011-03-25Z</start_date_time>
        <stop_date_time>2015-04-30Z</stop_date_time>
    </Time_Coordinates>
    <Primary_Result_Summary>
        <purpose>Science</purpose>
        <processing_level>Derived</processing_level>
    </Primary_Result_Summary>
    <Investigation_Area>
        <name>MESSENGER</name>
        <type>Mission</type>
        <Internal_Reference>
            <lid_reference>
                urn:nasa:pds:context:investigation:mission.messenger
            </lid_reference>
            <reference_type>collection_to_investigation</reference_type>
        </Internal_Reference>
    </Investigation_Area>
    <Observing_System>
        <name>MESSENGER</name>
        <Observing_System_Component>
            <name>MESSENGER</name>
            <type>Spacecraft</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:instrument_host:spacecraft.mess
                </lid_reference>
                <reference_type>is_instrument_host</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
         <Observing_System_Component>
            <name>NS</name>
            <type>Instrument</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:instrument:ns.mess
                </lid_reference>
                <reference_type>is_instrument</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
    </Observing_System>
```

```
        <Target_Identification>
            <name>Mercury</name>
            <type>Planet</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:target:planet.mercury
                </lid_reference>
                <reference_type>collection_to_target</reference_type>
            </Internal_Reference>
        </Target_Identification>
    </Context_Area>
```

M.1.3.3 Collection Area

The **Collection** area contains two attributes – **description**, which is optional, and **collection_type**, which is required. The **collection_type** attribute must have one of the following values:

- Browse
- Calibration
- Context
- Data
- Document
- Geometry
- Miscellaneous
- SPICE Kernel
- XML Schema

Here is the **Context_Area** part of our example Inventory label.

```
    <Collection>
        <collection_type>Data</collection_type>
    </Collection>
```

M.1.3.4 File_Area_Inventory

File_Area_Inventory is similar in structure to **File_Area_Observational** described in F.2.2, with constraints specific to inventory tables. It has a **File** subclass and an **Inventory** subclass. Both are required.

Here is the **File_Area_Inventory** part of our example Inventory label. It includes the required attributes and a few, but not all, of the optional attributes. In most labels **File_Area_Inventory** will look exactly like this except for the values of **file_name**, **creation_date_time**, and **records**.

```
    <File_Area_Inventory>
        <File>
            <file_name>collection_eetable_inventory.csv</file_name>
            <creation_date_time>2016-04-08</creation_date_time>
```

```
        </File>
        <Inventory>
            <offset unit="byte">0</offset>
            <parsing_standard_id>PDS DSV 1</parsing_standard_id>
            <records>3</records>
            <record_delimiter>Carriage-Return Line-Feed</record_delimiter>
            <field_delimiter>Comma</field_delimiter>
            <Record_Delimited>
                <fields>2</fields>
                <groups>0</groups>
                <Field_Delimited>
                    <name>Member Status</name>
                    <field_number>1</field_number>
                    <data_type>ASCII_String</data_type>
                    <maximum_field_length unit="byte">1</maximum_field_length>
                    <description>
                       P indicates primary member of the collection
                       S indicates secondary member of the collection
                    </description>
                </Field_Delimited>
                <Field_Delimited>
                    <name>LIDVID_LID</name>
                    <field_number>2</field_number>
                    <data_type>ASCII_LIDVID_LID</data_type>
                    <maximum_field_length unit="byte">255</maximum_field_length>
                    <description>
                        The LID or LIDVID of a product that is a member
                        of the collection.
                    </description>
                </Field_Delimited>
            </Record_Delimited>
            <reference_type>inventory_has_member_product</reference_type>
        </Inventory>
    </File_Area_Inventory>
```

For further discussion regarding collection products, see the *SBN PDS4 Wiki* or Section 9C of the *Standards Reference* [2].

M.2 Product_Bundle

Like collections, bundles are aggregate products. A bundle is a set of related collections. PDS does not impose requirements on bundles except that (1) bundle LIDs must be distinct within the overall holdings of PDS, and (2) each bundle must be approved by a PDS peer review. As part of the design discussion with your consulting DN data engineer (Section 4.3 of this document), you should have determined which collections will be included in your bundle.

A **Product_Bundle** identifies all of its member collections. Unlike collection products, which consist of a label and an inventory table, a bundle product consists of a label only. The inventory is embedded within the label because there are usually only a small number of collections in a bundle. A **Product_Bundle** may optionally refer to a separate **readme** file.

M.2.1 Generating and Populating a Product_Bundle Label

You may use the procedure described in Appendix E to create a draft **Product_Bundle** label and modify it using the information in this section. As an example, we will use the label for the bundle **izenberg_pdart_meap** in the online *DPH Examples*

(https://pds.nasa.gov/pds4/doc/examples/). It looks like this with the lower-level tags closed up. This bundle contains four collections.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.sch"
schematypens="http://purl.oclc.org/dsdl/schematron"?>

<Product_Bundle xmlns="http://pds.nasa.gov/pds4/pds/v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://pds.nasa.gov/pds4/pds/v1
https://pds.nasa.gov/pds4/pds/v1/PDS4_PDS_1700.xsd">
    <Identification_Area>
    <Context_Area>
    <Bundle>
    <Bundle_Member_Entry>
    <Bundle_Member_Entry>
    <Bundle_Member_Entry>
    <Bundle_Member_Entry>
</Product_Bundle>
```

M.2.1.1 Identification_Area

The **Identification_Area** class includes the same subclasses that it did in the **Product_Collection** label in Section M.1.3.1 above and in the **Product_Observational** label in Appendix F. The `<logical_identifier>` tag is filled in with the LID for the bundle as a whole (Section 5.2). The `<title>` also applies to the bundle as a whole. The `<Citation_Information>` and `<Modification_History>` tags are not required but are shown as examples. The description in `<Citation_Information>` should be a summary of the bundle contents suitable for display in a web browser.

Here is the **Identification_Area** part of our example **Product_Bundle** label.

```
    <Identification_Area>
        <logical_identifier>urn:nasa:pds:izenberg_pdart14_meap</logical_identifier>
        <version_id>1.0</version_id>
        <title>Izenberg PDART 2014 MESSENGER Advanced Products Bundle</title>
        <information_model_version>1.7.0.0</information_model_version>
        <product_class>Product_Bundle</product_class>
        <Citation_Information>
            <author_list>Izenberg, Noam</author_list>
            <publication_year>2016</publication_year>
            <description>
                MESSENGER Advanced Products (MEAP) by Noam Izenberg, PDART 2014
            </description>
        </Citation_Information>
        <Modification_History>
            <Modification_Detail>
                <modification_date>2017-02-09</modification_date>
                <version_id>1.0</version_id>
                <description>
                    Peer reviewed version of the Izenberg PDART 2014 MEAP bundle
                </description>
            </Modification_Detail>
        </Modification_History>
    </Identification_Area>
```

M.2.1.2 Context_Area

The **Context_Area** is optional in a **Product_Bundle** label, but its use is recommended because it contains attributes that are used in searching for PDS4 products. Filling in accurate values for these attributes will help users who are searching for data products like yours. The components of **Context_Area** are the same as those described above in Section M.1.3.2. They are all optional.

Here is the **Context_Area** part of our example **Product_Bundle** label.

```
<Context_Area>
    <Time_Coordinates>
        <start_date_time>2004-08-13Z</start_date_time>
        <stop_date_time>2015-04-30Z</stop_date_time>
    </Time_Coordinates>
    <Primary_Result_Summary>
        <purpose>Science</purpose>
        <processing_level>Derived</processing_level>
    </Primary_Result_Summary>
    <Investigation_Area>
        <name>MESSENGER</name>
        <type>Mission</type>
        <Internal_Reference>
            <lid_reference>
                urn:nasa:pds:context:investigation:mission.messenger
            </lid_reference>
            <reference_type>bundle_to_investigation</reference_type>
        </Internal_Reference>
    </Investigation_Area>
    <Observing_System>
        <name>MESSENGER</name>
        <Observing_System_Component>
            <name>MESSENGER</name>
            <type>Spacecraft</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:instrument_host:spacecraft.mess
                </lid_reference>
                <reference_type>is_instrument_host</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
        <Observing_System_Component>
            <name>MASCS</name>
            <type>Instrument</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:instrument:mascs.mess
                </lid_reference>
                <reference_type>is_instrument</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
        <Observing_System_Component>
            <name>GRS</name>
            <type>Instrument</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:instrument:grs.mess
                </lid_reference>
                <reference_type>is_instrument</reference_type>
            </Internal_Reference>
        </Observing_System_Component>
```

```
            <Observing_System_Component>
                <name>NS</name>
                <type>Instrument</type>
                <Internal_Reference>
                    <lid_reference>
                        urn:nasa:pds:context:instrument:ns.mess
                    </lid_reference>
                    <reference_type>is_instrument</reference_type>
                </Internal_Reference>
            </Observing_System_Component>
        </Observing_System>
        <Target_Identification>
            <name>Mercury</name>
            <type>Planet</type>
            <Internal_Reference>
                <lid_reference>
                    urn:nasa:pds:context:target:planet.mercury
                </lid_reference>
                <reference_type>data_to_target</reference_type>
            </Internal_Reference>
        </Target_Identification>
    </Context_Area>
```

M.2.1.3 Reference_List

The **Reference_List** class is optional in the **Product_Bundle** label. Here you may put any number of internal or external references relevant to the bundle. See Section 8, Documenting the Archive, for an explanation of internal and external references, and see Appendix F.2.6. for instructions on modifying **Reference_List**. Our example label omits this class.

M.2.1.4 Bundle

The **Bundle** class contains two attributes – **bundle_type** and **description**. If you are creating a bundle that includes a data collection, set **bundle_type** to "Archive". If there is no data collection, set **bundle_type** to "Supplemental". The **description** attribute is optional.

Here is the **Bundle** part of our example label.

```
    <Bundle>
        <bundle_type>Archive</bundle_type>
    </Bundle>
```

M.2.1.5 File_Area_Text

The **File_Area_Text** class is used if an optional **readme** file is included in the bundle. If there is no **readme** file, **File_Area_Text** should be omitted.[16]

Our example bundle in the *DPH Example Archives* does not have a **readme** file, but here is an example of how you would use **File_Area_Text** to describe a **readme** file:

```
    <File_Area_Text>
        <File>
            <file_name>readme.txt</file_name>
            <local_identifier>README.FILE</local_identifier>
```

[16] A **readme** file should give the reader a high-level overview of the bundle contents and structure.

```
            <creation_date_time>2010-03-12T11:59:04</creation_date_time>
            <file_size unit="byte">22875</file_size>
            <md5_checksum>5ef7af310b99d8189e670830c954a290</md5_checksum>
            <comment>Introduction to the bundle</comment>
        </File>
        <Stream_Text>
            <name>VG2 Jupiter plasma bundle</name>
            <local_identifier>BUNDLE.DESCRIPTION</local_identifier>
            <offset unit="byte">0</offset>
            <parsing_standard_id>7-Bit ASCII Text</parsing_standard_id>
            <description>
                Voyager 2 Jupiter plasma bundle description in ASCII text.
            </description>
            <record_delimiter>Carriage-Return Line-Feed</record_delimiter>
        </Stream_Text>
    </File_Area_Text>
```

File_Area_Text has two subclasses, both of which are required: **File** and **Stream_Text**.

The **File** subclass describes the actual readme file. Only the **file_name** attribute is required.

The **Stream_Text** subclass describes how the text in the readme file can be found and extracted. The **offset**, **parsing_standard_id**, and **record_delimiter** attributes are required. The **offset** attribute is the offset of the text from the beginning of the file (in bytes in this example). If the text starts at the beginning of the file, **offset** is 0. The **parsing_standard_id** attribute provides the name of the standard used for extracting the text from the byte stream in the file (see Table 4-1 for possible values). The **record_delimiter** attribute gives the delimiter that separate lines in the text. Currently the only acceptable value is "Carriage-Return Line-Feed".

M.2.1.6 Bundle_Member_Entry

The members of a bundle are specified using the **Bundle_Member_Entry** class. The **Bundle_Member_Entry** class is repeated for each collection product in the bundle. This class must occur at least once in the **Product_Bundle** label.

Our example bundle contains four collections, including the collection we labeled in Section M.1.3 above. Here are the **Bundle_Member_Entry** parts of our example label.

```
    <Bundle_Member_Entry>
        <lid_reference>
            urn:nasa:pds:izenberg_pdart14_meap:data_imagecube
        </lid_reference>
        <member_status>Primary</member_status>
        <reference_type>bundle_has_data_collection</reference_type>
    </Bundle_Member_Entry>
    <Bundle_Member_Entry>
        <lid_reference>
            urn:nasa:pds:izenberg_pdart14_meap:data_eetable
        </lid_reference>
        <member_status>Primary</member_status>
        <reference_type>bundle_has_data_collection</reference_type>
    </Bundle_Member_Entry>
    <Bundle_Member_Entry>
        <lid_reference>
            urn:nasa:pds:izenberg_pdart14_meap:data_tnmap
        </lid_reference>
        <member_status>Primary</member_status>
        <reference_type>bundle_has_data_collection</reference_type>
    </Bundle_Member_Entry>
```

```
<Bundle_Member_Entry>
    <lid_reference>
        urn:nasa:pds:izenberg_pdart14_meap:document
    </lid_reference>
    <member_status>Primary</member_status>
    <reference_type>bundle_has_document_collection</reference_type>
</Bundle_Member_Entry>
```

In the **Bundle_Member_Entry** class:

- You must use either the **lid_reference** or the **lidvid_reference** attribute for each member collection. Collections which are primary members of the bundle may be specified by either a LID or LIDVID depending on which is more appropriate for the particular bundle. Discuss which option to use with your consulting node.
- The value for **reference_type** depends on the type of collection. It must be one of the following:

Collection Type	Value for reference_type
Browse	bundle_has_browse_collection
Calibration	bundle_has_calibration_collection
Context	bundle_has_context_collection
Data	bundle_has_data_collection
Document	bundle_has_document_collection
Geometry	bundle_has_geometry_collection
Miscellaneous	bundle_has_member_collection
SPICE Kernel	bundle_has_spice_kernel_collection
Schema	bundle_has_schema_collection

- The value for the **member_status** attribute — "Primary" or "Secondary" — specifies whether the collection is a primary or secondary member of the bundle, respectively (see Section M.1.1).

The complete **Product_Bundle** label used in our example is online in the *DPH Examples* https://pds.nasa.gov/pds4/doc/examples/.

Change Log

Revision	Date	Description	Author
0.1	Mar 30, 2009	Initial draft based on information collected by the Data System Working Group.	R. Joyner
0.2	Aug 6, 2009	Updated versions of all Classes	R. Joyner
0.2.1	2010-08-31	Complete overhaul, but only partly successful through page 25	R. Simpson
0.22	Aug 31, 2010	Integrate Simpson and Joyner docs	R. Joyner etal
0.22.1	2010-09-22	Edits through Section 3, except Section 2	Simpson
0.22.2	2010-09-24	Added comments from Mitch Gordon	Simpson
0.22.2	2010-10-01	Significant edits to Section 4	Simpson
0.22.3	2010-10-25	Significant edits to Sections 1,3-6	Simpson
0.3	2011-04-15	Significant edits addressing Build 1b comments	M.Gordon etal
0.3.1	2011-04-21	Additional content added	M.Gordon etal
0.3.2	2011-05-02	Significant reorganization, additional edits	M.Gordon etal
0.3.3	2011-06-29	Make Sections 1-2 more user friendly	R. Simpson, M. Gordon
0.3.4	2011-09-06	Major changes Sections 2, 3, 8,14	M. Gordon, R. Joyner
0.3.5	2011-10-31	Updates for schema 0.5.00g	M. Gordon, R. Joyner
0.3.6	2012-01-31	Updates for schema 0.7.0.0.j	M. Gordon, R. Joyner
0.3.7	2012-03-28	Updates for schema 0.7.0.0.j	M. Gordon, R. Joyner
0.3.8	2012-05-28	Updates for schema 0.8.0.0.k	M. Gordon, R. Joyner
0.3.9	2012-10-01	Updates for schema 0.3.0.0a	M. Gordon, R. Joyner
0.3.10	2013-04-01	Major changes for schema 0.3.1.0b	M. Gordon, R. Joyner, R. Simpson

1.0.0	2013-05-01	Changes for compliance with schema 1.0.0.0	M. Gordon, R. Joyner, R. Simpson
1.2.0	2014-07-01	Changes for compliance with schema 1.2.0.0	M. Gordon, R. Joyner, R. Simpson
1.3.0	2014-09-01	Changes for compliance with schema 1.3.0.0	M. Gordon, R. Joyner, R. Simpson
1.4.0	2015-04-15	Changes for compliance with schema 1.4.0.0	M. Gordon, R. Joyner, R. Simpson
1.4.0	2015-06-28	Standardized use of "schema" and "Schematron files" in various places (such as footnote 2 in Section 1.3)	Simpson
1.4.0	2015-06-28	Moved [2] and [4] to Section 1.5.2; per the CCB charter and other documents, the SR and CD are as much controlling as the IM spec.	Simpson
1.4.0	2015-06-28	Changed "supporting node" to "consulting node" per terminology in *PDS4 Concepts*.	Simpson
1.4.0	2015-06-30	Simplified Section 4 by removing references to VG2 PLS example data.	Simpson
1.4.0	2015-07-02	Rewrote Appendix D, because it didn't work	Simpson
1.4.0	2015-07-25	Rewrote 6.1 to include product selection guidelines driven by CCB-111	Simpson
1.4.0	2015-08-10	Revised and simplified Section 7 for consistency across oXygen and Eclipse XAEs and PDS validation software.	Simpson
1.4.0	2015-08-20	Reorganized Sections 10-11 to consolidate topics and reduce repetition.	Simpson
1.4.0	2015-08-20	Streamined and reorganized Section 12. Expanded and corrected examples in 12.1. Removed most references to restricted documents from the original Section 12; then added 12.2 specifically about restricted documents.	Simpson
1.4.0	2015-08-27	Change "section" to upper case when referring to a numbered section.	Simpson
1.4.0	2015-08-28	Rewrote Section 3.0 to make it an introduction to labels and XML rather than a cookbook of how to manipulate templates (already in Section 7)	Simpson
1.4.0	2015-09-24	Added Appendices F, G, and H	Simpson

1.4.0	2015-10-06	Removed Section 14 and references thereto (the examples are outdated and have many errors). The section may be revised and returned in a later edition.	Simpson, R. Joyner
1.4.0	2015-10-06	There have been many other changes since DPH v1.3.0. The Change Log entries above should be considered representative. In many ways v1.4.0 is a new document.	Simpson. R.Joyner
1.4.1	2016-02-23	Changed date and version number on cover; updated Table of Contents.	Simpson
1.4.1	2016-02-23	Substituted period for comma at the bottom of page 13; corrected misspelling of "array" on page 18, line -6. Added missing close parenthesis in 'blue' text in the middle of page 23.	Simpson
1.4.1	2016-02-23	Replaced smart quotes by regular quotes in example labels.	Simpson
1.4.1	2016-02-23	Removed hyperlink notation from example labels	Simpson

Version 1.7.0

Version 1.7.0 of the Data Provider's Handbook incorporates changes made in the Information Model versions 1.5.0, 1.6.0, and 1.7.0. There are no corresponding versions 1.5.0 and 1.6.0 of the Data Provider's Handbook because of a delay in updating the document. To keep the Data Provider's Handbook version synchronized with the Information Model version, the Data Provider's Handbook version was incremented from 1.4.0 to 1.7.0.

Version 1.7.0 is a complete rewrite of the previous version. The sections and appendices have been re-ordered and much of the content revised, based on comments from PDS personnel and the judgment of the PDS Document Authors Team. The goal has been to make the flow of information more logical in order of presentation and in level of detail for the reader who is not a PDS expert.

Revision	Date	Description	Author
1.7.0	2016-12-08	Reorganization of sections and appendices	Slavney
1.7.0	2016-12-12	CCB-98: Choose most specific class available to describe an object. Added statement to end of section 6.3.1. to match SR section 9F.	Slavney
1.7.0	2016-12-13	CCB-146 and CCB-160: Added "ros" and "jaxa" as valid agencies in Appx K, Forming LIDs for Context Products.	Slavney

1.7.0	2016-12-15	Incorporated comments from Ed Guinness, Matt Tiscareno, Santa Martinez, Lev Nagdimunov, Dick Simpson, Ron Joyner, and Richard Chen	Slavney
1.7.0	2017-02-03	Revised text throughout	Slavney
1.7.0	2017-02-28	Revisions throughout, mostly minor	Simpson
1.7.0	2017-04-01	Revised appendices, except for Appendix K, Creating and Using Local Data Dictionaries.	Slavney