

PDS4 Concepts

1st Draft, edited 22 March 2011

2nd Draft, edited 20 May 2012

TBD:

Resolve speculative/uncertain/unfinished portions highlighted in red

Update to 0800j

Part I: Flesh out, reorder, and renumber the documents in Section 2.3

Part I: write (or remove) Section 2.4

Appendix D: Update view of dictionaries, list of terms, and definitions (from IM 0500g)

Contents

PDS4 Concepts	1
Contents.....	2
I. Introduction	5
1. Background	5
1.1 The Need for Standards	5
1.2 The Archiving Process	5
2. This Document	6
2.1 Outline.....	6
2.2 Audience	6
2.3 Related Documentation	6
2.4 Conventions	6
II. Overview of PDS4.....	7
3. Data Structures	7
3.1 Observational Data	7
3.2 Documents.....	7
3.3 Supplementary Information.....	8
4. XML and XML Schema.....	8
4.1 What is XML?	8
4.2 Why XML?	9
4.3 What is XML Schema?.....	9
4.4 Why XML Schema?.....	9
4.5 Terminology	10
5 Namespace.....	11
6. Data Dictionaries.....	12
6.1 What is a Data Dictionary?.....	12
6.2 PDS4 Data Dictionaries.....	12
6.3 Creating Data Dictionaries	13
6.4 Using Data Dictionaries.....	13
6.5 The PDS Dictionary Data Base.....	14

7. Labels	14
7.1 Organization.....	15
7.2 Creation.....	16
7.3 Validation	16
8. Archive Organization.....	17
8.1 Products	17
8.2 Collections.....	19
8.3 Members of Collections	19
8.4 Bundles.....	20
8.5 Logical Identifiers	20
9. Archiving Strategies	21
9.1 Preliminary Steps	21
9.2 Understand the Data.....	21
9.3 Design the Products	22
9.4 Design the Collections.....	22
9.5 Design the Bundles.....	23
9.6 Final Notes	23
III. PDS4 Data Structures	24
10. Scalars	24
10.1 Character Types	24
10.2 Binary Types	25
11. Arrays	25
11.1 Elements.....	25
11.2 Axes.....	26
11.3 Storage Order.....	26
11.4 Additional Considerations.....	26
12. Tables	27
12.1 Character Tables	27
12.2 Binary Tables	28
12.3 Records and Fields	29
13. Parsable Byte Streams	29
13.1 Flat Text.....	29

13.2 Flat UTF-8 Text	30
13.3 Delimited Tables	30
14 Encoded Byte Stream.....	31
14.1 PDF/A	31
14.2 Other Formats.....	31
Appendix A – Introduction to the PDS4 Information Model	33
A.1 Introduction	33
A.2 Building Blocks:.....	34
A.3 Related Definitions	34
A.4 Products.....	36
A.5 Collections and Bundles.....	37
A. 6 Delivery	37
A.7 XML.....	37
A.8 Management	38
Appendix B – PDS4 Glossary	39
Appendix C – PDS4 Acronyms and Abbreviations.....	45
Appendix D — PDS4 Data Dictionary	47
D.1 Introduction and Purpose.....	47
D.2 Related Documents	47
D.3 Terminology.....	47
D.4 PDS4 Data Dictionary Structure.....	48
D.5 Management of Attributes and Classes	53
D.6 Acknowledgements	54

I. Introduction

This *Concepts Document* provides overviews of PDS4, including an introduction to terminology needed for understanding and applying PDS4 standards to archives of planetary science data. The document itself does not define standards for PDS4; in cases of conflict, the definitive reference is [5].

1. Background

NASA's Planetary Data System (PDS) is a federation of 'discipline nodes,' each specializing in a subset of planetary science — geosciences, atmospheres, rings, etc. PDS has been operational for two decades, most of that time in its version 3 (PDS3). A lot has changed in those two decades; PDS version 4 (PDS4) has been developed to bring both the archiving process and use of the archived data into the modern era.

1.1 The Need for Standards

Standards lie at the heart of information transfer and interoperability. They provide the framework within which we browse the web, make online purchases, and even open and print this document. PDS standards ensure consistent description of planetary science data so that data providers, programmers, and end-users all know what to expect when creating and working with PDS files. Standards also guarantee that the data collected and archived today will still be readable and usable generations from now — a long-term return on today's investments made in flying planetary spacecraft and funding ground-based planetary research.

1.2 The Archiving Process

When a data provider first contacts PDS, one of the discipline nodes will be assigned as the principal, or consulting, node for the mission, instrument, or project. A large mission may have different consulting nodes for different groups of instruments; when this happens, one discipline node will be designated as the lead node.

The consulting node provides basic training, copies of standards documents and tutorials, development materials, and advice throughout the archive development effort. As the archive is being designed, the consulting node recommends additional information, over and above the observational data, which will be required — calibrations, documentation, etc. When data production starts, the consulting node provides validation support for standards compliance and conducts an external peer review of data submitted; it then monitors archive quality through final delivery.

More details about the discipline nodes, peer review process, and available tools can be found at <http://pds.nasa.gov>.

2. This Document

This document provides an overview of a PDS4 archive and describes the concepts behind the archive design and implementation.

2.1 Outline

After this Introduction, there are three more parts to this document. The next part provides overviews of data structures, the application of XML to PDS4, use of labels for describing products, organization of products into archives, and the steps that go into successful design of an archive. The third part provides more detail on structures and formats which appear in typical PDS4 archives. A set of appendices provides additional detail on selected topics.

2.2 Audience

This document is intended for a general audience with little or no background in PDS, XML or data preparation. It is appropriate for both data preparers and end users who wish to familiarize themselves with basic PDS4 requirements, structures, and definitions before diving into the full text of documents such as [1], [2], and user's manuals for PDS4 software tools.

2.3 Related Documentation

- [1] *PDS4 Standards Reference*, v.4.0.4, 10 April 2012.
 - [2] *PDS4 Data Provider's Handbook*, v.0.3.7, 28 March 2012
 - [3] Consultative Committee for Space Data Systems Secretariat, *Reference Model for an Open Archival Information System (OAIS)*, CCSDS 650.0-B-1 Blue Book, Washington, DC: National Aeronautics and Space Administration, January 2002.
 - [4] <http://www.wikipedia.org>
 - [5] *PDS4 Information Model*, v.0.7.0.0.j, 30 March 2012
- others

2.4 Conventions

TBW? Style conventions for the document. This needs some discussion, as the style is not as consistent as it should be.

II. Overview of PDS4

Part II of this document provides an overview of PDS4 data formatting standards and the terminology used in preparing data for archiving. The archiving process is actually fairly simple and straight-forward; it can be efficiently implemented as a ‘pipeline’ for many archives. However, documenting the file formats and context within which the data were acquired and processed is labor-intensive and must be done with both care and completeness.

3. Data Structures

PDS4 recognizes four ‘base’ data structures: array, table, parsable byte stream, and encoded byte stream. These data structures are tightly constrained, particularly for observational data. For most situations, simple formats for arrays and tables are sufficient. Where the simple formats cannot be applied, PDS has approved certain external standards — in many cases specific *subsets* of those external standards — as acceptable for its archival holdings.

3.1 Observational Data¹

The vast majority of observational data can be described as either N-dimensional arrays or fixed-width tables. Images, for example, are 2-dimensional arrays of identically formatted pixels. Fixed-width tables (text or binary) store data in records comprising fields of potentially different types but with the field assignments repeating in successive records. PDS4 data standards dictate what storage formats are acceptable for pixels in arrays and fields in tables.

3.2 Documents

Documentation includes any textual or text-based information supplied to assist the user in understanding, interpreting, calibrating, or otherwise manipulating the data. Frequently such documentation includes graphics and/or images to assist in comprehension. PDS has adopted the PDF/A² standard as the archival format for complex documents. UTF-8³

¹ PDS considers raw and partially or fully processed data derived from instrument (raw) data to be ‘observational.’ For example, both the raw frames and a geometrically and radiometrically calibrated mosaic of images would be ‘observational data.’

² “PDF/A” is a shorthand reference to “ISO 19005-1:2005— Document Management — Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1).” This standard, based on Adobe’s Portable Document Format reference version 1.4, is designed for use in archival storage of documents. It builds on the underlying PDF standard by adding additional requirements to ensure that the document file itself contains sufficient information that it can be

encoded text files may be used for simple documents that do not require graphics or special formatting control.

UTF-8 documents and ASCII spreadsheets are examples of PDS parsable byte streams; PDF/A and JPEG files are examples of PDS encoded byte streams.

3.3 Supplementary Information

Most submissions of observational data cannot stand alone – they require additional information to place the data in its appropriate context within a mission, observing campaign, or discipline. ‘Supplementary information’ and ‘supplementary data’ are terms applied generally to this additional archival material, which may include some or all of the following, and more:

- Browse products;
- Descriptions of spacecraft, instruments, observing geometry, etc.;
- Calibration data, including both observations and distilled results such as flat-field data, filter curves, transformation tables, etc.;
- Observing and command logs;
- Alternate formats for archival documents;

Some supplementary information, such as instrument descriptions, are essential to the long-term viability of an archive and will be required by the consulting node. Data providers are encouraged to include all other information that might be helpful to an end user.

Data structures for supplementary information/data are frequently in the form of images, tables, and documents. Exceptions include the alternate formats for archival documents, which may be based on PDS-approved external standards.

4. XML and XML Schema

The current implementation of the PDS4 system uses XML for all labels and inter-process communication. XML Schema (distinct from XML, and discussed more fully below) is used to define and constrain the structure and content of labels.

4.1 What is XML?

XML is the eXtensible Markup Language — a set of syntax rules that can be used in just about any application that involves parsing text. If you’ve seen HTML, then XML will have a familiar look; but XML is a much more formal syntax.

reproduced exactly from its contents. These requirements include font, graphics, and color management information.

³ UTF-8 is one of several multi-byte character encoding schemes that are used to implement Unicode. UTF-8 uses the minimum number of bytes required to represent each Unicode code point, so that the byte length of a single character may vary within a UTF-8 document. UTF-8 uses only one byte to encode the 7-bit ASCII code points, however.

4.2 Why XML?

Following its recommendation by the World Wide Web Consortium (W3C), XML has become an international standard for information interchange. It provides a fixed and well-defined syntax for creating document structures. As XML has increased in popularity, a wealth of third-party software has been developed to support the creation, parsing, and syntax validation of XML files (called ‘documents’ in XML parlance). PDS and its community can leverage this existing software not only for syntax validation — important enough in itself — but also to reduce the effort required to create new software to read, write, and manipulate the information in a variety of environments.

In addition, the namespace aspect of XML (and XML Schema, following) provides a convenient method for implementing local data dictionaries (LDDs), as described below. LDDs contain new definitions needed within specific disciplines or mission contexts so that data preparers can tailor their labels to the data in hand without impacting the more general structures defined at the ‘**master**’ data dictionary at the PDS level.

4.3 What is XML Schema?

XML is only useful as an interchange standard if both the sender and receiver know the definitions of the various tags⁴ used in the XML document. XML defines the syntax for the tags, but not their names or meanings. PDS4 uses XML Schema⁵ to define those tags and constrain their content.

XML Schema, also known as W3C XML Schema (WXS) and XML Schema Document (XSD), is a large and complex framework for defining the schema — the specific grammar and content structure — to be used in a set of XML documents. An XML Schema file is itself written in XML.

4.4 Why XML Schema?

XML Schema has the advantages of being widely supported, having a number of predefined data types useful to PDS, and offering support for the types of constraints and structure definitions required in PDS4 labels.

As with XML itself, there are many third-party tools that support the creation and application of XML Schema files, and the schema files created can be used to validate and verify the contents of the corresponding XML labels. Further, XML Schema files contain

⁴ In XML a tag is a character string delimited by “<” and “>”. An ‘XML element’ is a structure that begins with <tag>, contains ‘content,’ and ends with </tag>. For example, <date>2009</date> is an XML element establishing the date as 2009. For more information, including definitions of technical terms, see Appendices A and B.

⁵ *Schema* is a Greek word meaning *shape* or *plan*; its plural is *schemata* [4]. In English both *schemata* and *schemas* are used; PDS has adopted the latter.

detailed documentation of label structure and contents, which can be saved as part of the supplementary information for the data in the archive. In fact, PDS uses only a subset of the capabilities available in XML Schema.

PDS has developed procedures and tools to assist users in working with XML Schema to produce the schemas needed for their XML labels.

4.5 Terminology

Some basic XML terminology will be useful for talking about PDS XML files and their contents. Consider this snippet of XML, which might appear in a movie database application:

```
<movie>
  <title>Bedtime for Bonzo</title>
  <firstRelease>1951</firstRelease>
  <director>Frederick de Cordova</director>
  <screenplayBy>Lou Breslow</screenplayBy>
  <screenplayBy>Val Burton</screenplayBy>
  <storyBy>Ted Berkman</storyBy>
  <storyBy>Raphael Blau</storyBy>
  <starring>Ronald Reagan</starring>
  <starring>Diana Lynn</starring>
</movie>
```

In this example, `<movie>`, `<title>`, and `<storyBy>` are all tags. “Bedtime for Bonzo” is the content of the XML element `title`, and the content of the XML element `movie` is all of the elements (the tags with all their content) that come between `<movie>` and `</movie>`.

Because PDS data structures are based on an object-oriented design methodology, PDS uses the term *attribute* to refer to simple tags (like `title` and `director`) and the term *class* to refer to tags with complex content, like `movie`. In the example above, using PDS parlance, `title` and `starring` are *attributes* of the *class* `movie`.

In XML parlance, a file that contains well-formed (that is, syntactically correct) XML-formatted text is referred to as an *XML document*. An XML document has a single *root tag* – the first (and highest-level) XML element in the file, which contains all the other elements that comprise the document. PDS labels are well-formed XML documents.

An *XML attribute* is a qualifier on an XML element. For example, in

```
<length unit="foot">3</length>
```

`unit` qualifies `length`, so the content of the XML element should be interpreted as “3 feet.” PDS uses XML attributes in only a few situations, such as specifying units. The term

“XML attribute” will always be used in those cases — to distinguish *XML attribute* from the more common PDS use of *attribute*.

5 Namespace

A namespace is a context for defining something. For example, in the phrases “car title” and “movie title”, the word *title* has two very different meanings — “automobile ownership documents” or “name of the movie,” respectively. “Car” and “movie” provide contexts that have a significant effect on interpretation of “title.”

XML namespaces provide similar context for definitions. Two XML elements with the same name but different namespaces will (usually) have different definitions. The example above might look like this in XML:

```
<movie:title>...</movie:title>
<car:title>...</car:title>
```

The namespaces are *movie* and *car*, respectively. The value of `<movie:title>` will no doubt be a string, but the value of `<car:title>` is likely to be a set of subsidiary XML elements containing information such as the Vehicle Identification Number, make, model, year, registered owner, etc.⁶

Namespace has an analog in language. Using the notation above, `<French:journal>` means “newspaper” whereas `<English:journal>` is more akin to “magazine.” In this example the two namespaces — French and English — provide different contexts, which change the interpretation of the word.

Namespaces are managed by stewards; a single namespace can have only one steward, but a steward can manage many namespaces. The steward of the French language is the Académie Française, charged with forging a common French language in 1634; there is no comparable steward for English. In PDS there are different namespaces for common terms, discipline node terms, and mission terms; PDS assigns a steward to each.

⁶ Some additional detail: The `movie:` and `car:` prefixes in this example illustrate how namespaces are referenced in an actual XML file; but, in practice, these strings would more likely be nicknames (abbreviations, if you prefer) for the true namespace identifiers, which most often look like URLs. The association between the nicknames and the actual namespace identifiers is made at the top of the XML file. The specific nicknames used aren’t significant to XML, but choosing reasonable nicknames and using them consistently is very significant to human readers. In PDS4, XML label files will not only define these prefixes but also provide explicit pointers to the XML Schema files containing the element definitions for each namespace so that the label can be validated against the appropriate definitions.

6. Data Dictionaries

A data dictionary defines the meaning and structure of classes and attributes used in XML files; it can also constrain the content within an XML element. A one-to-one correspondence between namespaces and dictionaries is not required and is, in fact, the exception. However, it may be helpful to think of a data dictionary as containing all definitions within a single namespace in the sections below.

6.1 What is a Data Dictionary?

A data dictionary contains a list of terms and definitions used to describe one or more collections of data. Practically speaking, a data dictionary must be both human-readable and also structured in such a way that it can be used by a machine — to interpret the content of other XML files. A properly constructed data dictionary facilitates enforcement of syntax and semantic constraints placed on values of attributes. It also enables verification that class definitions have been properly implemented by giving the explicit list of attributes that define the class, indicating which are required, optional, and/or repeatable. A data dictionary also specifies whether a class definition must or may include associations — one-way relationships between the class being defined and another class (see Appendix A for more information).

A typical data dictionary will exist in several forms:

- an XML or XSD file created by a data preparer
- an XML Schema file used for validation of labels
- entries in an operational data base
- a printed document in which the definitions, specifications, and constraints have been abstracted to facilitate use by human developers

The XML/XSD file will typically be the ‘reference’ version of the data dictionary. Software utilities then generate the other forms by creating the XML Schema file (if not already available), updating the data base, and creating a printable version (such as a PDF/A).

6.2 PDS4 Data Dictionaries

PDS4 data dictionaries define classes and attributes used in PDS4 XML files by specifying tags, their meanings, and the acceptable values (including structure) that may appear as content. For example, the data dictionary that defines the `movie` class in Section 4.5 (above) must also define all of the classes and attributes within it, such as `<title>` and `<starring>`. Most of these will be simple text strings, but `<firstRelease>` is probably going to be an integer, constrained to have a value of 1878 or greater.

In PDS4 there is a PDS-wide data dictionary defining classes and attributes common to all labels. There are also node data dictionaries containing the classes and attributes specific

to each discipline node, and there may be a data dictionary for each data preparer working on each archiving project; these latter two categories are called 'local data dictionaries' (LDDs). To control the proliferation of data dictionaries, PDS lead nodes may insist that missions be served by a single LDD. Ultimately, all dictionaries will be loaded into the PDS4 dictionary data base (DDB), which integrates all dictionaries into a single master data base that users may query. Since the discipline node, mission, and other stewards have carefully grouped their class and attribute definitions into distinct namespaces, even two uses of "title" or "journal" (for example) coming from the same data preparer will be separable.

6.3 Creating Data Dictionaries

Note: This section is almost entirely speculative – the details of how namespace dictionaries will be created and what will/won't be allowed in them is in active discussion.

PDS personnel compile data dictionaries that include terms common to the system or to a discipline node. Individual data preparers create mission, instrument, and other LDDs, often coordinated through the consulting and/or lead node. A 'registration authority,' such as PDS, manages dictionary development by establishing and coordinating stewards and their namespaces within its domain.

There are two approaches to creating a data dictionary file for a PDS4 archive; both use resources available from PDS:

- Use the Dictionary Creation Toolkit (DCT)
- Create the dictionary directly using the Dictionary Schema Template (DST).

The DCT contains a dictionary creation schema, PDS schemas containing type parameters and unit lists, a utility for converting a dictionary XML file into the schema files needed for creating and validating labels, and a utility for submitting the LDD to the PDS DDB. The DCT guides the data preparer through attributes and classes for all likely product labels. It is particularly suited for developers with no previous XML Schema experience.

The data preparer who wants finer control or who wants to take advantage of more advanced XML Schema features (which may be out of reach because of default settings in the DCT) may wish to create a data dictionary directly from the PDS4 DST. The consulting node can then provide assistance for uploading the completed template and integrating it with the DDB.

6.4 Using Data Dictionaries

The typical PDS product label will reference at least one (and possibly several) different data dictionary schema files; each of the schema files may have multiple namespaces. The most likely three (if they are maintained separately) will be:

- The PDS global data dictionary, which contains the definitions of the globally-required classes and attributes (like identifiers) as well as the fundamental data structures (arrays, tables, documents, etc.);
- The node data dictionary supplied by the consulting PDS node (or, possibly, several nodes), which defines attributes and classes needed by users of that node for searching or manipulating the data; and
- The local data dictionary created by the data preparer — usually one per mission — to define the attributes and classes used to provide detailed documentation of the product contents.

A few words TBW on how the references are implemented.

6.5 The PDS Dictionary Data Base

This section can't be completed until the DDB interfaces are defined.

7. Labels

Once you have the PDS global data dictionary (and, possibly, node and mission dictionaries), you're ready to construct labels for the products in your archive. PDS4 labels are XML files that describe the contents of one or more files (observational data, document files, calibration tables, etc.). With one important exception (discussed in Section 8.4) the PDS label is always in a file separate from the 'data' it describes. This means you don't have to tinker with your 'data' files. As long as your image, table, document, or other file can be described by a PDS4 label your efforts can be limited to constructing labels, not reorganizing 'data' files.

The PDS label plus the data are called a 'product.' [The *formal* definition of 'product' is based on 'objects' rather than files, and not all objects are in the PDS; these and other subtleties are discussed in Appendix A.]

PDS4 labels are derived from one of several master schemas, which are maintained by the PDS Engineering Node and contain all of the current data dictionary information. As noted above, **the master schemas include contributions from discipline nodes, missions, and others.** Sections of the master schema not relevant to the product can be removed using an editor; the result becomes the schema for the product in question — sometimes called a 'tailored' or 'specific' schema, depending on the degree of editing. If desired, the schema can be converted to a label template. Either the schema or the template can then be inserted into a production pipeline for automatic label generation. Or the schema or template can be hand-edited if the number of labels is small. 'Tailored' schemas and/or templates are often created by staff at the consulting node, then passed to the data preparer who does final editing, pipeline integration, etc. A 'specific' schema is created by the person or group doing the archiving.

7.1 Organization

A PDS label serves several purposes, including:

- Carrying unique identification for its product within the PDS (think serial number);
- Pointing to the physical files which are the actual product content (*e.g.*, data);
- Describing the structure of the component files (*e.g.*, table rows and columns)
- Documenting other aspects of the product — observing conditions or processing status for observational data; or authorship, abstract, and copyright information for documents.

These purposes map loosely into ‘areas’ within a label:

- XML Header — contains links to local and remote schemas
- Identification Area
 - Information which uniquely identifies the product
- Observation Area
 - Descriptions of the specific observation, laboratory experiment, etc.
 - Subsections for information appropriate to
 - one or more discipline nodes, and
 - the mission or an equivalent entity
- Reference List Area
 - Information on other products, journal articles, etc., relevant to understanding this product (both internal and external to PDS)
- File Area
 - Pointers to the component file(s), and
 - Detailed descriptions of the file(s)

The label is an XML document (see example outline below). Immediately after the XML header (lines 1-2) is a root tag, `<Product_Observational>`, which, in conjunction with its `</Product_Observational>`, envelopes everything else in the label. Inside this XML element, information is organized into classes with each of the ‘areas’ above being represented (except Reference List Area, which is optional). Classes contain attributes or other classes, and the hierarchy continues downward to the lowest level of attributes — attributes of the binary fields, in this example. Note that for each `<tag>` there is always a matching `</tag>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://pds.jpl.nasa.gov/schemas/PDS4_PDS_0700j.sch"?>
<Product_Observational ...">
  <Identification_Area>
    ...
  </Identification_Area>
  <Observation_Area>
    ...
  </Observation_Area>
  <File_Area_Observational>
    <File>
      ...
    </File>
  </File_Area_Observational>
</Product_Observational>
```

```

    ...
    <Table_Binary>
      ...
      <Record_Binary>
        ...
        <Field_Binary>
          ...
        </Field_Binary>
      </Record_Binary>
    </Table_Binary>
  </File_Area_Observational>
</Product_Observational>

```

7.2 Creation

As noted above, labels can be created within processing pipelines by using either schemas or templates. If we zoom in on the `Table_Binary` definition in the example above, we see that its *template* has variables that need to be supplied for each table generated. In particular, the number of records changes from table to table. If the pipeline executes a search-and-replace for `$records` in the template (line 4), the correct value can be easily inserted. Other variables, such as start and stop times in `Observation_Area` can be inserted in the same way.

```

<Table_Binary>
  <local_identifier>RSR_TABLE</local_identifier>
  <offset>0</offset>
  <records> $records </records>
  <encoding_type>BINARY</encoding_type>
  <Record_Binary>
    <record_length>8260</record_length>
    <fields>72</fields>

    <Field_Binary>
      <name>SFDU CONTROL AUTHORITY</name>
      <field_number>1</field_number>
      <field_location>1</field_location>
      <data_type>ASCII_Short_String_Collapsed</data_type>
      <field_length>4</field_length>
      <description>
        An ASCII string giving the SFDU Control Authority
        for this data type. Set to 'NJPL', meaning the
        data description information for this type of SFDU
        is maintained by the NASA/JPL Control Authority.
      </description>
    </Field_Binary>
  ...

```

Software experts may prefer to use schemas with pipelines rather than templates, but the result is the same. For every product, the values are inserted and the correct label is generated for its respective product.

7.3 Validation

Finished labels must be validated against any schemas upon which they were based, such as the original 'master' schema and the 'tailored' schema received from the consulting node. This validation can be done inside an XML editor, through programming libraries in languages such as Perl and Java, with command-line utilities such as Linux *xmllint*, or with combinations of these. Data preparers should adopt appropriate validation procedures early so that errors do not persist and propagate. Labels that fail validation when they

reach PDS will be rejected. Consulting PDS nodes can provide help in finding and using validation tools.

8. Archive Organization

The word ‘archive’ is used generally to mean large collections of data being held primarily for preservation. **When referring to the entirety of the PDS holdings, the word is often capitalized** (Figure 8.1). In this document, we use ‘archive’ to mean “the entire collection of data, documents, and supplementary information created by a data preparer.” So, for example, when we talk about a ‘mission archive,’ we mean all the files coming from a single mission.

Because the typical archive usually contains thousands, if not hundreds of thousands of files, some organization is required just to ensure proper storage and retrieval. This section describes the hierarchy used to organize archives created by data preparers both physically and logically. Where Figure 8-1 shows the organization from the top down; here we describe the construction of an archive from the bottom up.

8.1 Products

In PDS we are primarily concerned with digital ‘objects;’ examples include 2-dimensional images and ASCII tables. These are usually stored in files, but image and table files sometimes begin with ‘headers’ — which provide summaries or serve as ‘fingerprints’ and have completely different structures. Hence, there can be two digital objects in such image and table files; we need to describe and track both. We create a ‘description object’ to describe each digital object — one for the data and another for the header (if any). The concatenation of description objects (plus a little overhead required by XML) becomes our PDS label; the label plus all of the objects it describes is called a ‘product’ (see Appendix A for more details).

Less commonly, the digital objects, which need to be tracked together as a single product, are spread over multiple files. For example, red, green, and blue arrays that form a color image could be stored in separate files and labeled and tracked as a single product (they could also be three separate products). Note that, although a single label may describe multiple objects/files, any individual object may be described by only one label. For simplicity, PDS also requires that files containing multiple objects be associated with only one label. You can’t have objects from two or more products in the same file.

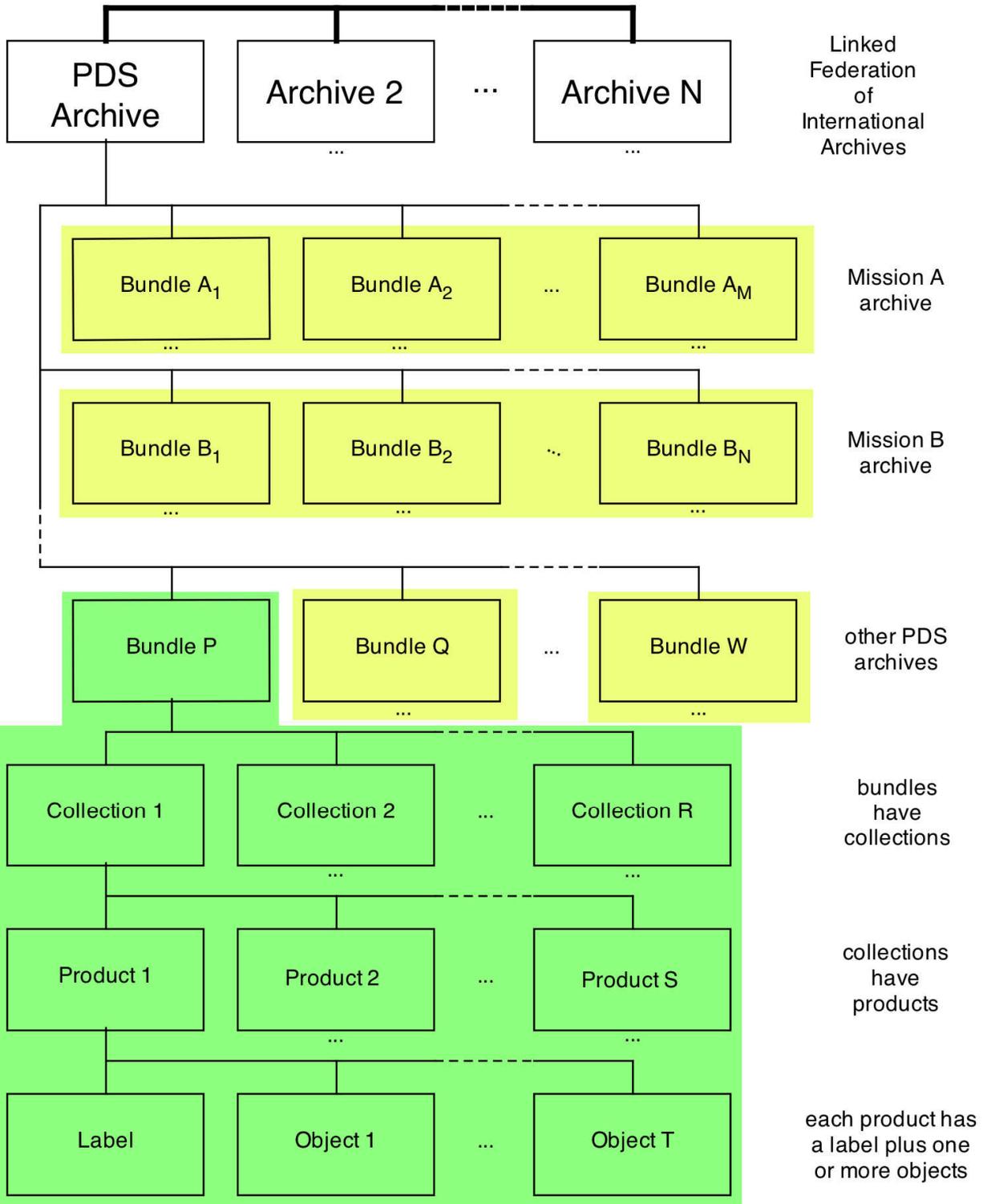


Figure 8 – Organization of archives, including links from PDS to other archive systems, **which interact by exchanging XML documents**. Some missions (A and B) may create multiple bundles. A single bundle may have several collections, each of which has several products comprising a label and one or more objects (green)

The data files referenced by a product label must either be in the same directory as the label, or in a subdirectory of the directory containing the label. Every product label contains a 'logical identifier' for the product, and that logical identifier is unique across the entire PDS. Since interoperability across the international federation of archives (Figure 8-1) requires that each member system be independently identified, uniqueness across PDS also guarantees uniqueness across the federation (see Section 8.5).

8.2 Collections

Products are grouped into 'collections' by type and content. Observational data, for example, may be gathered into an observational data collection, documents into a document collection, and supplementary data into a supplementary collection. Most archives are large enough that there likely will be many observational data collections, and possibly several document and supplementary collections. **Observational data collections are typically split along instrument, instrument mode, and/or data type** (e.g., image vs. spectra) boundaries. The products included in a single collection should be closely related; the criteria for 'close' are usually left to the data provider working with the consulting node.

To specify a collection, the data preparer creates a table, which lists all of the PDS *label* files and their logical identifiers that meet the criteria for being 'closely related.' The data preparer then creates a label for the table file. So a collection can be thought of as a higher-order product — one that aggregates products that share one or more characteristics. The collection label, like any other label, contains a logical identifier that is unique across the entire PDS. Formally speaking, a PDS collection is only the table and its label; but it is common, when discussing collections, to think of them as the table, label, and all of the member products.

The products listed in the collection table must be physically located in the same directory as the collection label, in one or more subdirectories below it, or somewhere else. Handling this third case requires a brief digression.

8.3 Members of Collections

A 'basic product' is the simplest product in PDS4; it is one or more data objects (and their label), which constitute (typically) a single observation, document, etc. The only PDS4 products that are *not* basic products are `Product_Collection` (the collection table and its label, described above) and `Product_Bundle` (the corresponding product at the bundle level, which will be described below). Every basic product must be a 'primary member' of one (and only one) collection. Basic products may be 'secondary members' of any number of collections. In practice, this means that every new product enters PDS as a primary member of one collection; then its identifier may be re-used, showing that it is a secondary member of other collections. For example, a collection of images of the Saturn system may be subdivided into Saturn atmosphere, Saturn ring, and Saturn satellite collections without requiring the copying of the image products themselves. The images would be 'basic products' and 'primary members' of

the original collection and ‘secondary members’ (as appropriate) of the other collections. The atmosphere, ring, and satellite collections would be nothing more than their respective inventory tables (and labels). Mixing primary and secondary members in a single collection is not prohibited; but data preparers should discuss possible repercussions with the consulting node before exercising the option.

8.4 Bundles

Collections are themselves gathered into bundles. Small archives may have a single bundle containing all of the collections. Larger archives will be broken into bundles along some convenient lines (mission phase, source instrument, review schedule, etc.).

To specify a bundle, the data preparer creates a `Product_Bundle` label, which is like other product labels (Section 7.1) except that it includes a `Bundle_Member_Entry` for each member collection instead of a File Area. This is the one type of label (mentioned in Section 7) that incorporates the ‘data’ into the label itself, although it does so using the class-attribute structure. Each `Product_Bundle` label includes an Identification Area with a logical identifier which is unique across PDS.

The collections listed in the bundle label must be physically located in one or more subdirectories of the directory containing the bundle label. **[Note: This constraint needs to be discussed further by DDWG, as is it not clear that it will be possible – or reasonable to enforce – in all cases.]**

8.5 Logical Identifiers

Every product, collection and bundle in an archive will have a unique logical identifier (LID). For example, this is the LID for an image (an observational data product) from the fictional Sunburn mission, where fields are delimited by the colon (“:”) character:

```
URN:NASA:PDS:Sunburn-1:SolCam-Cr1-Raw:Image01321
```

The `URN:NASA:PDS` fields are required in all PDS LIDs. They identify the LID as a PDS system reference. The next field, `Sunburn-1`, is the ID for one of the bundles produced by the Sunburn mission – the one containing this data product. All of the collections in this bundle will have the bundle ID as part of their LIDs. The next field, `SolCam-Cr1-Raw`, is the ID of the particular collection to which the product belongs. All products in this collection will have this collection ID as part of their LIDs. Finally, the `Image01321` field is a product ID, which is unique within the `SolCam-Cr1-Raw` collection. By appending the bundle and collection IDs to the product ID, data preparers can be sure of creating product identifiers that will be unique across the entire PDS; with the `URN:NASA:PDS:` prefix, each product has a unique LID across the entire universe of federated archive organizations.

Note that the LID or logical identifier is not tied to the physical location of the data – thus it is a logical identifier, as opposed to a file specification. We use LID or logical identifier in this document whenever we mean the full LID, from URN: to the last data preparer-specified segment. We use the term ID to refer to a segment within a LID (*i.e.*, a string between colons); for example, in the LID above, the product ID is Image01321 and the collection ID is SolCam-CR1-Raw.

9. Archiving Strategies

The ultimate goal of each archiving effort should be to create one or more usable bundles that will stand on its (their) own for generations. That is, 50+ years from now a user should be able to read, understand, and use the data in the archive despite having no access to the original source of the observations or the team that prepared them.

9.1 Preliminary Steps

Before anything is known about the data, the prospective data preparer can do a few things to prepare for the archiving effort:

- Gain some familiarity with XML. Your PDS consulting node can supply a working label and schema examples to play with if you want to dive directly into the PDS end of the pool; but any introductory tutorial on XML syntax will also provide a good basis for PDS work with XML.
- Find an XML editor. There are shareware and commercial editors available for all major platforms. Your PDS consulting node can suggest one or two to check out if you don't have a local XML guru to advise you. In particular, familiarize yourself with the validation capabilities of your editor.
- Gain some familiarity with PDS data. Search the current holdings for data similar to those you plan to prepare, and see what sorts of data structures and additional documentation are provided above the minimal PDS requirements.

9.2 Understand the Data

For each instrument producing observational data, determine what array and/or table structures best match the data. If none, you may need help from your consulting node to get started. Consider what additional information should be included in the product labels to document the observations. For large missions, consider what attributes should be defined uniformly across instruments and observations; make plans with other mission data preparers to agree on definitions and value formats early in the design process and make plans for developing a mission data dictionary.

Also think about the supplementary products that should be provided with the observational data, including documentation, calibrations, browse directories, observation logs, command sequences, and look-up tables.

Finally, consider what higher-order and derived products might be produced.

9.3 Design the Products

The choice of what constitutes a single product is usually pretty clear. A user's guide (perhaps in multiple formats) is a single document product; a filter response curve is a single supplementary product; and so on. Sometimes the distinction for observational data is not quite so clear. An image and a corresponding bad pixel map can be labeled as a single product or as two separate products, for example. For the most part, PDS will accept any reasonable grouping suggested by the data preparer unless past experience indicates that a proposed choice represents a future problem for end-users. If that seems likely, your PDS consulting node will suggest an alternate organization that is more likely to satisfy end-users and reviewers.

While documents and supplementary products frequently have one-of-a-kind labels created by a data preparer using an XML editor, observational product labels are usually produced by a pipeline. Observational products also generally require a significant amount of very specific documentation in the labels to describe observing conditions, instrument performance, processing history, targeting information, and so on.

Only the most general characteristics appear in the PDS global classes; **those and the node-level classes will be incorporated into the tailored templates or schemas created by the consulting node for the data preparer as starting points.** So the major work of the data preparer in designing product labels is in determining what additional mission- and instrument-specific information should be added to schemas or templates and defining the corresponding attributes and classes needed in the local data dictionary.

The schemas and/or label templates can then be passed to the pipeline programmers for creating the PDS labels.

9.4 Design the Collections

Closely related products are gathered into collections. The major design question for collections is simply deciding which products are to be included in what collections. PDS requires that collections contain products that are 'closely related,' a criterion that is usually left to the data preparer (working with the consulting node) for definition. The vast majority of data preparers divide data into collections based on things like mission phase, target, instrument, data processing level, and so on. Collections should aggregate products in a way that makes sense from an end-user point of view. For example, one collection might contain all raw images recorded by a navigational camera during the cruise phase of

its mission, another might have calibrated images from the same phase, and a third might have raw images from the encounter phase of the mission.

9.5 Design the Bundles

Collections are gathered into bundles mainly to facilitate data management and accounting. Consequently, the aggregation criteria for bundles vary more widely than for collections. Examples include:

- All collections from a mission, mission phase, or time period
- All collections related to a particular target
- All collections produced by a specific instrument

Any other similarly reasonable criterion would also suffice for designating bundle contents. Once again, your PDS consulting node will be glad to advise.

9.6 Final Notes

Missions and other large data producers obligated to archive with PDS will generally create a document explaining the archive design, designating responsible parties, and laying out a schedule for reviews and deliveries. This document might be called an archive plan, an interface control document, a system interface specification, or something similar. It will generally require sign-off from both the data provider and the PDS. Your PDS consulting node can provide examples of existing documents for you to follow. Whether you include this document in the archive depends on whether a future data user would find it helpful.

Smaller data production efforts often follow a less formal set of steps, although peer review and archiving requirements are just as rigorous. For these smaller archives, the PDS-supplied tailored schemas may be all that is needed to complete the archiving process, and the node itself may incorporate the data into an existing bundle for permanent archiving.

Your PDS consulting node is there to answer any and all questions you might have about how to label and organize your data. Please make contact early and often during the design process to get answers to questions, suggestions for best practices, and working examples.

III. PDS4 Data Structures

In Part III we expand on some of the topics introduced in Section 3, focusing on the ‘data’ objects, how they are described, and constraints imposed by PDS.

The structure of the data may have already been fixed when the data arrive at the desk of the data preparer (archivist). If those formats are PDS-compliant, nothing more needs to be done with the data. If the data are not compliant, adjustments will be required. As a general rule, PDS does not recommend reformatting raw data because errors, however unlikely, may render the entire archive unusable in ways that can never be corrected. Reformatting more highly processed data is encouraged if it makes the resulting products more attractive to end users. In either case, designing data processing software with an eye toward the final archive can save both time and effort in the long run.

10. Scalars

Scalars are the smallest storage units in the PDS4 system. A single element of an array is a scalar, as is a single field in a table. PDS tightly constrains what scalar types may be used in tables and arrays to ensure long-term stability in the archive and to provide a sound basis for PDS and third-party software development.

10.1 Character Types

NOTE: DDWG still needs to discuss this material and finalize decisions on details such as:

- *ASCII vs. UTF-8 in strings or text that use character data;*
- *patterns for data types with restricted ASCII character sets;*
- *whether non-printing characters are allowed in strings (e.g., ASCII_Short_String_Collapsed is limited only to “ASCII”);*
- *circumstances under which 8-bit ASCII might be allowed (as opposed to 7-bit)*
- *how to express the ASCII (and UTF-8) characters TAB, SEMICOLON, and CARRIAGE-RETURN (among others) in documentation such as this*

Character fields in observational data files are restricted to the printable 7-bit ASCII characters, plus the blank character. Within this character set, PDS defines a number of specific data types based on formatting constraints and requirements; examples include ASCII_MD5_Checksum, which is limited to the characters 0–9, a–f, and A–F. These data types may be used in both binary and character tables; they are the *only* choices for character tables. *PDS also allows the non-printing TAB as a field delimiter in the Table_Delimited class and the NEW_LINE, and CARRIAGE_RETURN characters as record delimiters.* The full list of character data types is below:

ASCII_AnyURI	ASCII_NonNegative_Integer
ASCII_Boolean	ASCII_Numeric_Base16
ASCII_DOI	ASCII_Numeric_Base2
ASCII_Date	ASCII_Numeric_Base8
ASCII_Date_DOY	ASCII_Real
ASCII_Date_Time	ASCII_Short_String_Collapsed
ASCII_Date_Time_DOY	ASCII_Short_String_Preserved
ASCII_Date_Time_UTC	ASCII_Text_Collapsed
ASCII_Date_Time_YMD	ASCII_Text_Preserved
ASCII_Date_YMD	ASCII_Time
ASCII_Directory_Path_Name	ASCII_VID
ASCII_Identifier	Float3Vector
ASCII_Integer	UTF8_Short_String_Collapsed
ASCII_LID	UTF8_Short_String_Preserved
ASCII_LIDVID	UTF8_Text_Preserved
ASCII_MD5_Checksum	

10.2 Binary Types

Binary data types must be used in arrays (character arrays are prohibited in PDS) and in binary tables. The acceptable binary data types are:

ComplexLSB16	SignedLSB8
ComplexLSB8	SignedMSB2
ComplexMSB16	SignedLMB4
ComplexMSB8	SignedLMB8
IEEE754LSBDouble	UnsignedBitString
IEEE754LSBSingle	UnsignedByte
IEEE754MSBDouble	UnsignedLSB2
IEEE754MSBSingle	UnsignedLSB4
SignedBitString	UnsignedLSB8
SignedByte	UnsignedMSB2
SignedLSB2	UnsignedLMB4
SignedLSB4	UnsignedLMB8

11. Arrays

The simplest structure available for data storage is the (homogeneous) array — that is, an N-dimensional structure in which all the elements are the same data type. This is the structure underlying all images, spectral cubes, maps, and similar structures. To define an array you must first define the element type, and then define the number and lengths of each of the axes. Although N-dimensional structures are allowed in PDS, only $N \leq 3$ arrays have been fully developed as of this writing.

11.1 Elements

Any of the binary types listed in section 10.2 may be used as the data type for an array element. In addition, elements may be associated with a number of characteristics like units of measure, offsets, and scaling factors. The data preparer can also designate specific flag values to be used — for example, to indicate saturated pixels in an image. In the PDS4 XML label, the element and its attributes will be defined using the `Element_Array` class in the global PDS data dictionary.

11.2 Axes

Each array will have one or more axes. In the PDS4 label, each axis will be defined using the `Axis_Array` class in the global PDS data dictionary so that interpretive attributes (like the name of the axis) can be specified for each axis individually. The principal attribute of an axis is its length.

The order in which the axes are defined is significant; it is important to know which axis comes first, which second, *etc.*, so that axes can be accurately mapped to subscripts in storage arrays in program memory and to storage order in the physical file. In the PDS4 label, axis order will be stated explicitly using a sequence number in each axis definition. If you think of the axes of an array as corresponding to subscripts, then the axis with sequence number 1 would be the first (leftmost) subscript, axis number 2 the next, and so on.

11.3 Storage Order

The individual bits of any array element are stored in the order dictated by their scalar type; for example, LSB and MSB have opposite byte orders.

The elements of the array are stored such that the subscript along axis 1 (notationally, the leftmost axis) varies most rapidly; that along axis 2 next most rapidly; and so on. This means that the elements along the highest-numbered (rightmost) axis will be stored contiguously. This storage order has also been called column-major order historically, from 2-dimensional arrays in which the subscripts were thought of as [row, column] — though this becomes a less intuitive description as the number of axes increases.

This order also corresponds to the storage order required by the FITS standard, and it is the default storage order for 2-dimensional images under the PDS3 standard.

11.4 Additional Considerations

The mapping of logical array space to storage space can be tricky — as when going from an analysis format to a PDS4 archival format. It is absolutely essential that it be done correctly, though, so that high-level descriptive parameters can be properly applied. A two-dimensional image incorrectly stored will ultimately appear inverted on a display device, making it impossible for a user to determine the direction of celestial north relative to targets in the image, for example.

12. Tables

NOTE: This section may need to be modified depending on the outcome of discussions regarding UTF-8 vs. ASCII in data files. It is currently written assuming only ASCII will be allowed in character tables. We also need to decide

- 1) whether we want to allow columns which are vectors (i.e., corresponding to PDS3 ITEMS), and what the record delimiter will be;*
- 2) whether to require that all bits in a binary record be accounted for;*
- 3) whether to define a special data type to use for spare bits/bytes in binary files;*
- 4) whether enclosing quotes are counted toward field_length and field_location*
- 5) whether one-byte record delimiters are allowed (currently true)*

And, of course, if we prefer to address the four basic storage structures in this section, then this may need to be rewritten.

Tables — more specifically fixed-width tables⁷ — consist of a repeating record structure with each record comprising a set of fields. PDS requires that each field be a fixed-width scalar of an appropriate type. All records in the table must have exactly the same structure; consequently, all records in a table will have the same length.

Fixed-width tables come in two types — binary and character. Binary tables may contain character fields, but character tables may never contain binary fields.

12.1 Character Tables

Each field in a character table must have one of the data types listed in section 10.1. Each record in a character table must end with a **one- or two-byte** record delimiter consisting of the ASCII carriage return character and/or the ASCII line feed character.

Character tables will usually have ‘gutter space’ between fields. In most cases this will be a blank character; but it can also be a fixed-length string of blanks, a comma, or any other string of one or more valid characters (printing characters plus the blank). Quotes, which enclose a field of characters, are considered to be part of the gutter space. Gutter space is not required; but character tables without gutter space tend to be less readable.

Gutter space, where it exists, and the **one- or two-byte** record delimiter are considered to be outside the actual fields; their bytes should never be included in any of the field definitions in the character table record. So, if you add up the widths of all the defined fields, the difference between that sum and the total record length should be the number of bytes occupied by the gutter space and the record delimiters.

Character tables have two major advantages over binary tables for data storage:

⁷ Fixed-width tables are distinguished from tables with records of variable length — the `Table_Delimited` class (see Section 13.3)

- They are directly human-readable (or printable); and
- They can represent numerical values to arbitrary precision.

The human-readability factor makes character tables extremely stable for long-term storage. It is nearly impossible to misinterpret a character file as long as you're using the defined standard encoding. ASCII is one of the longest-lived computing standards in existence; since it is preserved in Unicode and UTF-8, it does not appear to be in any danger of becoming obsolete soon.

Because numeric values expressed as ASCII strings are not constrained by the limitations of binary conversion or byte counts, character tables can potentially contain integer values greater than those in binary integers and real values with more precision than is possible in binary floating point formats. Of course, the creation and manipulation of such values usually requires that they be represented in a binary format; but, if an observer creates hyper-precision values (for example the first 1000 digits of π) and wants them preserved, a character table may be the easiest option.

Despite the fact that character files generally require more bytes for storage than binary files, character files are often preferred by users because of their human-readability, the possibility of being able to read the files directly into spreadsheet programs, and the simplicity of reading the values using every major programming language without having to worry about detailed architectural issues like byte order on the target machine.

12.2 Binary Tables

Unlike character tables, binary tables have no record delimiters and no gutter space *per se*; if a data preparer generates them, they simply become additional binary fields. In fact, any character table can also be defined as a binary table (the converse is not true).

It is not necessary to define all binary fields in a PDS label; 'spares', the delimiters and gutter space already noted, and 'unknown' fields are not uncommon in binary files from some sources. The data preparer should describe all of the fields in a binary record that are known; there may, for example, be some fields in legacy data that are simply not documented. The data preparer should certainly describe all fields that are of interest to the investigation for which the data were collected.

The fields in a binary file will mainly be of the types listed in Section 10.2, though some of the character types listed in Section 10.1 may also appear. Dates and times, simple character strings, and flags (for example 'T' and 'F') are among the most common character fields in binary tables.

Binary tables have two major advantages over character tables for data storage:

1. For tables containing primarily numeric data, the binary version will likely be substantially smaller than the character equivalent; and

2. There is no conversion error in reading binary floating point numbers into memory on machines that use the same floating point storage format as PDS4.

You should check with your consulting PDS node about user preferences for binary versus character tables. Many users are happy to have a human-readable table that is an order of magnitude larger than a binary table simply because of the readability factor.

The IEEE-954 floating point format required by PDS4 for real (and complex) numbers is common across computing platforms as of this writing. For the average user who prefers binary, the major considerations are byte-ordering and byte-alignment.

12.3 Records and Fields

In PDS4 labels describing tables, the table is a class, which has records. The record is a class, which has fields. The field is a class, which has at least attributes and possibly more classes (see Sections 7.1 and 7.2). The specific field attributes vary somewhat depending on the type of table, record, and field; but they generally include name, data type, start position within the record, length, and description. Additional optional attributes (especially for numeric fields) include things like format, value offset, scaling factors, and unit. Classes that may appear within a field include `Special_Constants`, which provides a set of values to indicate special cases (missing, invalid, saturated, etc.), and `Packed_Data_Fields`, which allows binary parsing at the bit level.

13. Parsable Byte Streams

Parsable byte streams can be quickly and easily interpreted using standard rules. Common examples of parsable byte streams include:

- text stored in records representing lines on a printed page
- ASCII tabular data stored as records of variable length

In these two examples, each record would end with a delimiter (for example, an ASCII carriage-return line-feed pair), making the byte stream parsable. **PDS recognizes SPICE kernels, certain header structures, and XML files as falling within the definition of a parsable byte stream.**

13.1 Flat Text

Flat text (`Stream_Text` class) is the simplest parsable byte stream. It has only three required attributes:

- `offset` — displacement in bytes of the start of the text from the beginning of the parent structure (often set to “0”)

- `encoding_type` — set to `CHARACTER`, which allows UTF-8 and its subset ASCII
- `external_standard_id` — the non-PDS standard which makes the stream parsable (Would acceptable values include ASCII or UTF-8?)

Note that line delimiters (as mentioned for the example above) are only required if mandated by the external standard. Single-file ASCII documents without figures or special formatting meet the definition of parsable byte streams.

13.2 Flat UTF-8 Text

In this section we elaborate on UTF-8, a specific implementation of the Unicode character standard. UTF-8 text is the same as flat text (Section 13.1) so far as data structures are concerned; this section merely provides more background on UTF-8 and Unicode.

The Unicode standard assigns a unique number, called a code point, to every character, glyph and control code used in text files. A Unicode Translation Format (UTF) maps Unicode code points to a specific byte value sequence. There are several UTFs in use, including the most popular variants UTF-8, UTF-16 and UTF-32. PDS has designated UTF-8 as the text standard for PDS4.

Unicode allows the use of letters with diacritical marks as well as many single-character symbols (degrees, the Angstrom symbol, Greek letters, etc.), which are useful in text documents describing an archive. UTF-8 has several additional features that make it an appealing choice for PDS4 text files:

- It is completely backwards-compatible with 7-bit ASCII (the PDS3 text standard). That is, any existing flat text file that contains only 7-bit ASCII characters is automatically UTF-8 compliant.
- It uses only as many bytes as are actually needed to express the Unicode code point, so that the vast majority of characters used in PDS4 text files will require only a single byte representation. (UTF-16, as a counter-example, uses two bytes for all characters, which would effectively double the size of simple PDS4 text files.)
- It is not byte-order dependent. UTF-8 treats multi-byte characters as strings, so issues of byte storage order (LSB vs. MSB) do not affect UTF-8 strings.
- Conversion from native text to UTF-8 is supported in a wide variety of editors on all major platforms.

13.3 Delimited Tables

In contrast to fixed-width tables (Section 12), delimited tables (`Table_Delimited` class) have records of variable lengths. This structure allows more efficient storage when field values are expressed with varying precisions and, especially, if many fields are empty.

Delimited tables are also readily ingested into many commercially available spreadsheet programs⁸.

The number of fields and their content definitions do not change from record to record, but the format within each field is allowed to vary. Fields are separated by field delimiters (PDS allows the ASCII horizontal tab, comma, semi-colon, and vertical bar characters as field delimiters), and records are separated by record delimiters (ASCII carriage-return, line-feed, or both).

A delimited table which has four records delimited by carriage-return line-feed pairs and in which only one of 10 comma-separated fields is populated might look like this:

```
1 , , , , , , , , , , <CR><LF>
, 2 , , , , , , , , , , <CR><LF>
, , , 404 , , , , , , , , , , <CR><LF>
, , , , , , 7.797 , , , , , , , , , , <CR><LF>
```

14 Encoded Byte Stream

An encoded byte stream can only be interpreted after it has been ‘decoded’ according to some well known standard. This generally implies that computation is needed before the information can be seen and used. For this reason, and the related concern about long-term viability of encoded structures, PDS strictly limits use of encoded byte streams within archives.

14.1 PDF/A

All documents in PDS archives must appear in PDF/A (an encoded byte stream), UTF-8 (a parsable byte stream), or both.

PDF/A is an ISO standard format based on the PDF 1.4 definition. It is the only archival format that accommodates inline graphics and images. It requires that all fonts used in a document be embedded in the PDF, so that they will always be available for printing or displaying the document. Data preparers who will be producing PDF/A documents should, therefore, be careful to avoid proprietary and copyrighted fonts.

PDF/A is becoming an increasingly common output option for documents composed in the major commercial editors. PDS will accept either Level A or Level B compliance (you might see these listed as PDF/A-1a and PDF/A-1b in your format menu).

14.2 Other Formats

⁸ In PDS3, Table_Delimited was known as the SPREADSHEET

PDF, Microsoft Word, and Postscript are acceptable secondary formats for documents once a PDF/A or UTF-8 version has been included.

GIF, JPEG, PNG, and TIFF are among formats allowed for supplementary encoded images, such as in a browse collection.

Appendix A – Introduction to the PDS4 Information Model

The following sections define terms used across the Planetary Data System in its version 4 (PDS4). Definitions can be found where words are highlighted in **bold** font. Definitions of the most fundamental terms are given first in a ‘functional’ order — the order in which they might be needed to understand PDS4 from the ground up. *All* definitions are given alphabetically after this Introduction. Where there is a divergence between PDS4 definitions and those used in the Open Archival Information System Reference Model (OAIS RM) [1]—the basis for the PDS4 Information Model (IM)—there is a note. The PDS4 IM is the framework within which PDS4 has been designed and built. A few words have different meanings depending on the communities in which they are used; we have adopted modifiers if one use is clearly secondary in the PDS context. For example, ‘attribute’ is widely used in PDS, while ‘XML attribute’ has a different meaning, appropriate to an XML context.

A.1 Introduction

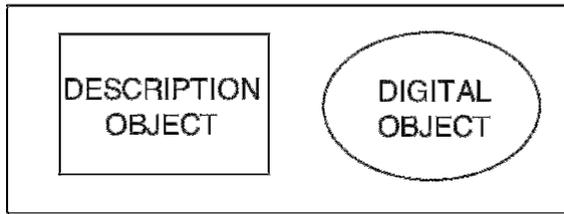
Our environment is filled with ‘objects’; some are tangible (like rocks and water), others are intangible (like beliefs and ideas). We call these ‘physical objects’ and ‘conceptual objects’, respectively. Since this is a glossary for the Planetary Data System, we also identify ‘digital objects’, which are the bits in a computer or storage device that can be interpreted and used — for example, as an image or a table of numbers. Bringing all three together: definitions of rocks and water are conceptual, instances or realizations of rocks and water are physical, and computer files containing photographs of rocks and water are digital. *For purposes of the PDS4 Information Model*, we call all three types (physical, conceptual, and digital) ‘**data objects**’. This allows us to manage all three using the same terminology and procedures even though physical and conceptual objects have no bits.

In order to *use* an object, we need information on both its structure and its meaning — in the case of digital objects, this is easy: how are the bits organized into rows and columns (for example) and what do they represent? We can fit both the object *and* its description into the archive. We can’t fit rocks, water, beliefs, or ideas into PDS4; but we can fit their descriptions. We can pair each description of a physical or conceptual object with its (imaginary) data object; then we can operate on these ‘pairs’ in exactly the same way we operate on the digital objects and their descriptions—except that nothing happens when we ask PDS4 to display the real rock, water, belief, or idea on a computer screen.

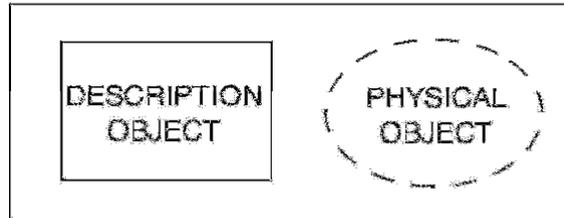
A data object paired with its description is called an ‘**information object**’ in the OAIS RM and in the PDS4 IM (Figure A-1). PDS4 is based entirely on ‘information objects’; in the XML implementation of PDS4, there is a little added XML window dressing (overhead), but the PDS4 information objects overwhelmingly dominate.

In the next section we start creating PDS4 from the ground up, constructing basic building blocks and defining terms.

INFORMATION OBJECT



INFORMATION OBJECT



INFORMATION OBJECT

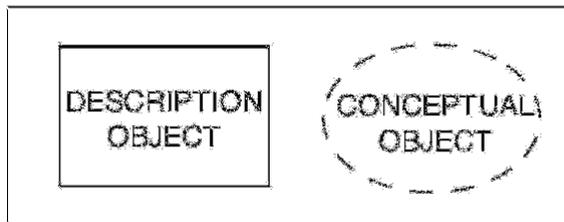


Figure A-1. The ‘information object’ is the basic building block of PDS4. It is the pairing of a description with a digital, physical, or conceptual ‘data object’. The fact that physical and conceptual data objects do not actually exist in the archive (dashed outlines) is not important so far as the PDS4 Information Model is concerned.

A.2 Building Blocks:

To build a functional data system, we require that every component be modeled — given a formal structure that explains content and constrains the operations that can be performed on it (and which it can perform). For PDS4 modeling an object oriented approach and vocabulary has been adopted. We start by defining ‘attribute’ and ‘class’.

attribute: A property or characteristic that provides a unit of information. For example, ‘color’ and ‘length’ are possible attributes.

class: The set of attributes (including a name) which defines a family. A class is generic — a template from which individual members of the family may be constructed. If the class ‘rope’ (its name) is defined by attributes ‘color’ and ‘length’, we can construct a family of ropes — *e.g.*, red and 3 m long, red and 4 m long, blue and 2 m long, ...

A.3 Related Definitions

association: An attribute that establishes a unidirectional relationship between two classes. For example, a table has records; ‘has record’ is the relationship between one entity (the table) and another (a record).

meta-attribute: An attribute of an attribute — that is, a ‘dictionary’ attribute, which is used to define one or more attributes in the PDS4 Information Model. For example, ‘conceptual_domain’ and ‘maximum_value’ are used in defining some attributes.

object: The realization of a single member of a family defined by a class. If the class ‘rope’ has attributes ‘color’ and ‘length’, we can construct a ‘rope’ family with three members — red and 3 m long, red and 4 m long, and blue and 2 m long. Each member is an object.

As noted above, there are three subclasses of object in PDS4: physical, conceptual, and digital; but only digital objects will be found in the digital archive. [Note: the OAIS RM calls these ‘data objects’]. We can now define the three subclasses:

digital object: An object which is real data — for example, a binary image of a redwood tree or an ASCII table of atmospheric composition versus altitude.

physical object: An object which is physical or tangible (and, therefore, does not itself fit into a digital archive). Examples of ‘physical objects’ include the planet Saturn and the Venus Express magnetometer. Note that an ASCII file describing Saturn is a *digital* object, not a physical object (nor a component of a physical object).

conceptual object: An object which is intangible (and, because it is intangible, does not fit into a digital archive). Examples of ‘conceptual objects’ include the Cassini mission and NASA’s strategic plan for solar system exploration. Note that a PDF describing the Cassini mission is a *digital* object, not a conceptual object (nor a component of a conceptual object).

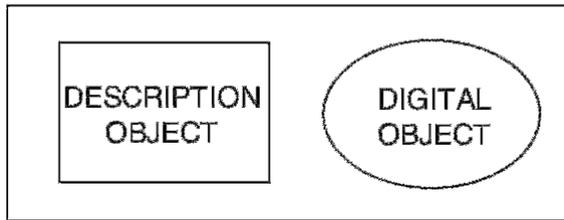
As noted above, objects, by themselves, are not very useful. We know nothing about their structures and have no clues as to how their contents should be retrieved and interpreted.

description object: Something that describes an object. As appropriate, it will have structural and descriptive components. Technically speaking, a ‘description object’ in PDS4 is a ‘digital object’ — a string of bits; but we assume that we can read it and, on that basis, give it a special name.

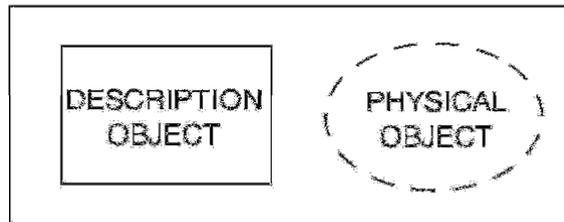
tagged digital object: A digital object paired with its companion description object (Figure A-2). [Note: In the OAIS RM this pair is known as an ‘information object’]

tagged non-digital object: A physical object or a conceptual object paired with its companion description object (Figure A-2). [Note: In the OAIS RM this pair is known as an ‘information object’]

TAGGED DIGITAL OBJECT



TAGGED NON-DIGITAL OBJECT



TAGGED NON-DIGITAL OBJECT

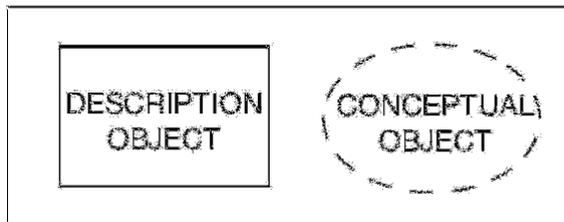


Figure A-2. 'Tagged digital objects' and 'tagged non-digital objects' are the fundamental components of PDS4 'products'. Dashed lines denote objects that are not actually in the archive. [Note: Tagged objects are 'information objects' in OAIS RM, per Figure A-1.]

A.4 Products

label: The aggregation of one or more description objects such that the aggregation describes a single PDS product. In the PDS4 implementation, labels are constructed using XML, which imposes a small amount of overhead.

identifier: A unique character string by which a product, object, or other entity may be identified and located. Identifiers can be global, in which case they are unique across all of PDS (and its federation partners). A local identifier must be unique within a label.

product: One or more tagged objects (digital, non-digital, or both) grouped together and having a single PDS-unique identifier. In the PDS4 implementation, the descriptions are combined into a single XML label. Although it may be possible to locate individual objects within PDS (and to find specific bit strings within digital objects), PDS4 defines 'products' to be the smallest granular unit of addressable data within its complete holdings.

basic product: The simplest product in PDS4; one or more data objects (and their description objects), which constitute (typically) a single observation, document, etc. The only PDS4

products that are *not* basic products are Product_Collection and Product_Bundle. Every basic product must be a primary member of one (and only one) collection. Basic products may be secondary members of any number of collections.

A.5 Collections and Bundles

collection: A list of basic products, all of which are closely related in some way. A collection is itself a product (because it is simply a list, with its label); but it is not a *basic* product.

primary member: A *basic product* is a primary member of the collection within which it first enters PDS4. Every basic product must be a primary member of one (and only one) collection.

secondary member: A *basic product* may be a secondary member of any number of collections. A collection which lists references to basic products already registered in PDS would identify those products as its secondary members. For example, if all Voyager images were in one primary collection, an analyst could define a new (subset) collection containing images which had Saturn's rings within the field of view; each of those image products would be a secondary member of the new collection.

bundle: A list of collections. For example, a bundle could list a collection of raw data obtained by an instrument during its mission lifetime, a collection of the calibration products associated with the instrument, and a collection of all documentation relevant to the first two collections.

A.6 Delivery

container: The physical equivalent of a package; the product manifest and all related files wrapped together for transfer — for example, in a ZIP, GZIP, or TAR file.

package: A product manifest and all related files *logically* grouped together for transfer.

A.7 XML

tag: Fundamental syntax in XML; a tag is a character string delimited by “<” and “>”. For example “<date>” is a tag.

XML element: An XML structure that begins with <tag>, contains ‘content’, and ends with </tag>. For example, “<date>2009</date>” is an XML element establishing the date as 2009. The allowed ‘content’ is specified in the PDS4 IM, which is propagated to the PDS4 Data Dictionary.

XML attribute: An attribute-value pair that is inserted into an XML element to provide additional information, such as units; the value is always quoted. For example

<date unit="year">2009</date>

XML schema: The definition of an XML document, specifying required and optional XML elements, their order, and parent-child relationships.

A.8 Management

registry: A data base that provides services for sharing content and metadata.

registration authority: An organization responsible for maintaining a registry — in this case, the PDS4 Information Model and its components. The registration authority for the Planetary Data System is ‘PDS’.

steward: A person or organization that manages a set of registered attributes and classes, typically as an agent for another or others. A registration authority must have at least one steward; it may have many. Stewards for PDS4 include PDS, the discipline nodes, and any mission wishing to conform to the PDS4 Information Model.

namespace: A context for defining classes and attributes. Two items with the same name but from different namespaces generally have different definitions. For example, “title” has a very different meaning in a *movie* namespace compared with its meaning in an *automobile* namespace.

Appendix B – PDS4 Glossary

archive: a place in which public records or historical documents are preserved; also the material preserved -- often used in plural. Sometimes capitalized when referring to all of PDS holdings — the PDS Archive.

array: an N-dimensional data structure in which every element has an identical data type. For example, a structure with 5 rows and 3 columns in which each element is a 2-byte signed integer would be an array.

archive bundle: Definition pending ... The Information Model definition “The Archive Bundle class describes a collection of collections” is unsatisfactory.

association: An attribute that establishes a unidirectional relationship between two classes. For example, a table has records; ‘has record’ is the relationship between one entity (the table) and another (a record).

attribute: A property or characteristic that provides a unit of information. For example, ‘color’ and ‘length’ are possible attributes.

basic product: The simplest product in PDS4; one or more data objects (and their description objects), which constitute (typically) a single observation, document, etc. The only PDS4 products that are *not* basic products are Product_Collection and Product_Bundle. Every basic product must be a primary member of one (and only one) collection. Basic products may be secondary members of any number of collections.

bundle: A list of collections. For example, a bundle could list a collection of raw data obtained by an instrument during its mission lifetime, a collection of the calibration products associated with the instrument, and a collection of all documentation relevant to the first two collections.

cardinality: The number of values allowed to an attribute or association in a single class. Cardinality in general is stated as a range with a minimum and maximum. For example, an *optional* attribute that may be multi-valued will have a cardinality of "0..*". A cardinality where the minimum and maximum are the same is often shown as the single value; for example, an attribute required to have exactly one value will have a cardinality of "1". When a value is required, the minimum cardinality is at least 1.

class: The set of attributes (including a name) which defines a family. A class is generic — a template from which individual members of the family may be constructed. If the class ‘rope’ (its name) is defined by attributes ‘color’ and ‘length’, we can construct a family of ropes — *e.g.*, red and 3 m long, red and 4 m long, blue and 2 m long, ...

class hierarchy: An ordering of classes which shows parent-child relationships.

collection: A list of basic products, all of which are closely related in some way. A collection is itself a product (because it is simply a list, with its label); but it is not a *basic* product.

collection inventory: Definition pending ... A PDS collection *is* the inventory.

data dictionary: a repository for definitions of classes and attributes

data model: Definition pending ... same as information model?

data object: A physical, conceptual, or digital object.

data product: Definition pending ... same as product?

data structure: a particular way of storing data in a computer so that it can be used efficiently

digital object: An object which is real data — for example, a binary image of a redwood tree or an ASCII table of atmospheric composition versus altitude.

encoded byte stream: a byte stream that may only be interpreted after it has been ‘decoded’ according to some well known standard

encoded structure: Definition pending ... same as “encoded byte stream”?

entity: Something that has a distinct, separate existence.

extension: In PDS4 a class extension is a subclass. Subclasses are more specialized versions of a class. They inherit attributes and behaviors from their parent classes, and they can have attributes of their own. For example, Array_2D is a PDS4 subclass of Array_Base.

homogeneous array structure: Definition pending ... by PDS definition, an array is both a structure and homogeneous.

identifier: A unique character string by which a product, object, or other entity may be identified and located. Identifiers can be global, in which case they are unique across all of PDS (and its federation partners). A local identifier must be unique within a label.

information model: a representation of concepts, relationships, constraints, rules, and [operations](#) to specify [data semantics](#) for a chosen domain of discourse. Specifically, the PDS Information Model (IM) is the representation that specifies PDS4.

information object: A data object paired with its description

inventory (see **collection inventory**): an itemized list of current assets or holdings

label: The aggregation of one or more description objects such that the aggregation describes a single PDS product. In the PDS4 implementation, labels are constructed using XML, which imposes a small amount of overhead.

label template: A text file which serves as a pattern for constructing labels

local: within a single label.

local dictionary: Definition pending ... same as LDD?

local data dictionary (LDD): a data dictionary for classes and attributes which are not defined across the entire PDS. Examples include data dictionaries for discipline nodes, missions, and individual archiving projects.

logical identifier (LID): An identifier which identifies the set of all versions of an object

logical identifier with version identifier (LIDVID): concatenation of a logical identifier with a version identifier

manifest: a list of contents

meta-attribute: An attribute of an attribute — that is, a ‘dictionary’ attribute, which is used to define one or more attributes in the PDS4 Information Model. For example, ‘conceptual_domain’ and ‘maximum_value’ are used in defining some attributes.

metadata: Data about data — for example, a ‘description object’ contains information (metadata) about an ‘object.’

mission: a task with which a group of people have been charged, usually by a government agency and including priority (if not exclusive) use of one or more spacecraft (see Investigation_Area type) ***

mission area: that part of a label which is specified by a mission

model: A representation or description designed to show an entity and its composition.

namespace: A context for defining classes and attributes. Two items with the same name but from different namespaces generally have different definitions. For example, “title” has a very different meaning in a *movie* namespace compared with its meaning in an *automobile* namespace.

node area: that part of a label which is specified by a PDS Discipline Node

object: The realization of a single member of a family defined by a class. If the class ‘rope’ has attributes ‘color’ and ‘length’, we can construct a ‘rope’ family with three members — red and 3 m long, red and 4 m long, and blue and 2 m long. Each member is an object.

object-description pair: Definition pending ... same as information object?

observational data: raw measurements from one or more instruments, or the results from processing such raw measurements.

observing campaign: an observational assignment with which a group of people have been charged (sometimes voluntarily) which extends over some period of time and which can be accomplished without significant construction of new equipment. (see Investigation_Area type) ***

parsable byte stream: a byte stream which can be parsed with standard rules — e.g., comma separated entries or standard punctuation; ‘decoding software’ is not needed.

parsable byte structure: Definition pending ... same as parsable byte stream?

PDS: see PDS4 Acronyms (below)

PDS attribute: Definition pending ... same as attribute? Not the same as XML attribute.

PDS4: see PDS4 Acronyms (below)

primary member: A *basic product* is a primary member of the collection within which it first enters PDS4. Every basic product must be a primary member of one (and only one) collection.

product: One or more tagged objects (digital, non-digital, or both) grouped together and having a single PDS-unique identifier. In the PDS4 implementation, the descriptions are combined into a single XML label. Although it may be possible to locate individual objects within PDS (and to find specific bit strings within digital objects), PDS4 defines ‘products’ to be the smallest granular unit of addressable data within its complete holdings.

product type: Definition pending ... nothing in the Information Model provides help. The class `Primary_Result_Description` has a type, but the enumerated values are all over the map and include disciplines, targets, instrument parts, and adjectives.

registry: A data base that provides services for sharing content and metadata.

registration authority: An organization responsible for maintaining a registry — in this case, the PDS4 Information Model and its components. The registration authority for the Planetary Data System is ‘PDS’.

repeating record structure: Tentative definition ... the part of a table which has successive records meeting the same structural definition

repository: a place, room, or container where something is deposited or stored (often for safety or preservation)

resource: The target (referent) of any Uniform Resource Identifier; the thing to which a URI points.

restored data: Definition pending ... data which have been recovered from storage and successfully prepared for archive in PDS

schema: a structural definition given in a formal language which serves as a blueprint for construction.

secondary member: A *basic product* may be a secondary member of any number of collections. A collection which lists references to basic products already registered in PDS would identify those products as its secondary members. For example, if all Voyager images were in one primary collection, an analyst could define a new (subset) collection containing images which had Saturn's rings within the field of view; each of those image products would be a secondary member of the new collection.

steward: A person or organization that manages a set of registered attributes and classes, typically as an agent for another or others. A registration authority must have at least one steward; it may have many. Stewards for PDS4 include PDS, the discipline nodes, and any mission wishing to conform to the PDS4 Information Model.

subclass: See extension.

supplementary data: Tentative definition ... additional archival material which is useful in understanding observational data; examples include: browse products, descriptions of instruments and other facilities important to data acquisition, information about observing geometry, calibrations, and observing and command logs.

table: a two-dimensional data structure composed of records, which themselves are heterogeneous but which repeat throughout the table. For example, a table could have 20 ASCII records, each of which has a 10-character date field, a comma, an 8-character time field, a comma, a 3-digit integer temperature field, and a 'new-line' character.

tag: Fundamental syntax in XML; a tag is a character string delimited by "<" and ">". For example "<date>" is a tag.

uniform resource name (URN): see PDS4 Acronyms (below)

version identifier (VID): an identifier which identifies the version of something else

XML: see PDS4 Acronyms (below)

XML attribute: An attribute-value pair that is inserted into an XML element to provide additional information, such as units; the value is always quoted. For example
`<date unit="year">2009</date>`

XML aware editor: use “XML editor” instead

XML document: a file that contains syntactically correct XML-formatted text

XML editor: an editor, which has special features allowing XML tag completion, XML validation, etc.

XML label: a label written using XML

XML root tag: the first (and highest-level) XML element in an XML document

XML schema: The definition of an XML document, specifying required and optional XML elements, their order, and parent-child relationships.

XML tag: same as tag.

XML template: A text file which serves as a pattern for constructing XML documents

XML element: An XML structure that begins with `<tag>`, contains ‘content’, and ends with `</tag>`. For example, “`<date>2009</date>`” is an XML element establishing the date as 2009. The allowed ‘content’ is specified in the PDS4 IM, which is propagated to the PDS4 Data Dictionary.

Appendix C – PDS4 Acronyms and Abbreviations

ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
COSPAR	Committee on Space Research
DDB	Dictionary Data Base
DCT	Dictionary Creation Tool
DDB	Data Dictionary Data Base
DPH	Data Provider's Handbook
DST	Dictionary Schema Template
ebXML	electronic business eXtensible Markup Language
ESA	European Space Agency
HTML	Hypertext Markup Language
IAG	International Association of Geodesy
IAU	International Astronomical Union
IEEE	Institute of Electrical and Electronics Engineers
IM	Information Model
ISO	International Standards Organization
ISO/IEC	International Standards Organization / International Electrotechnical Commission
ISO/TS	International Standards Organization / Technical Standard
JPL	Jet Propulsion Laboratory
LDD	Local Data Dictionary
LSB	Least Significant Bit
MD5	Message-Digest algorithm 5
MSB	Most Significant Bit
NAIF	Navigation and Ancillary Information Facility
NASA	National Aeronautics and Space Administration
NBS	National Bureau of Standards
OAIS	Open Archival Information System
PDF	Portable Document Format
PDS	Planetary Data System
PDS4	Planetary Data System, version 4
PSA	(ESA) Planetary Science Archive
PSDD	Planetary Science Data Dictionary
RM	Reference Model
SPICE	Spacecraft, Planet, Instrument, C-matrix (pointing), and Events kernels
SR	(PDS) Standards Reference
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTF	Unicode Transformation Format
W3C	World Wide Web Consortium
WXS	W3C XML Schema
XML	eXtensible Markup Language

XSD XML Schema Document

Appendix D — PDS4 Data Dictionary

NB: This appendix was written for Information Model 0.5.0.0.g; once the specification for data dictionary entries stabilizes, it will be updated to the latest IM.

D.1 Introduction and Purpose

The Planetary Data System (PDS) PDS4 Data Dictionary (DD) is an adjunct to the PDS4 Information Model (IM); together they define the organization and components of PDS4 product labels. The components of a product label are description objects created from classes and their attributes. This document provides an overview of the Data Dictionary, its management, and its use.

The PDS4 Data Dictionary is actually a data base; it is distributed to users in two versions — abridged and unabridged. The unabridged version includes an entry for every attribute in each class it appears. Since many attributes are used with several classes and the meaning often doesn't change, there is considerable repetition. The abridged version has been abstracted from the unabridged version; it contains full definitions but fewer details and none of the repetition. 'Housekeeping' information is suppressed in both versions.

D.2 Related Documents

- Controlling Documents

- PDS4 Information Model Specification — Version 0.5.0.0.g (19 October 2011 20:31:53 PDT) — The source for PDS4 class, attribute, and data type definitions.
- ISO/IEC 11179:3 Registry Metamodel and Basic Attributes Specification, 2003 - The reference schema for the PDS4 data dictionary.

- Reference Documents

- Glossary of PDS4 Terms - The source for terms used across the Planetary Data System in its version 4 (PDS4).
- PDS3 Planetary Science Data Dictionary - The online version of the PDS3 data dictionary was used as the source for a few data entries carried over to the PDS4 system.

D.3 Terminology

The following are definitions of some important terms used in the Data Dictionary; these are taken verbatim from the PDS4 Glossary.

1. **attribute:** A property or characteristic that provides a unit of information. For example, 'color' and 'length' are possible attributes.

2. **class:** The set of attributes (including a name) which defines a family. A class is generic — a template from which individual members of the family may be constructed. If a class named ‘rope’ is defined by attributes ‘color’ and ‘length’, we can construct a family of ropes — e.g., red and 3 m long, red and 4 m long, blue and 2 m long, ...
3. **association:** An attribute that establishes a unidirectional relationship between two classes. For example, a table has records; ‘has record’ is the relationship between one entity (Table_Base, the simplest table in PDS4 nomenclature) and another (Table_Record).
4. **object:** The realization of a single member of a family defined by a class. If the class ‘rope’ has attributes ‘color’ and ‘length’, we can construct a ‘rope’ family with three members — red and 3 m long, red and 4 m long, and blue and 2 m long. Each member is an object.
5. **conceptual object:** An object which is intangible (and, because it is intangible, does not appear in a digital archive). Examples of ‘conceptual objects’ include the Cassini mission and NASA’s strategic plan for solar system exploration. Note that a PDF describing the Cassini mission is a digital object, not a conceptual object (nor a component of a conceptual object).
6. **digital object:** An object which is real data — for example, a binary image of a redwood tree or an ASCII table of atmospheric composition versus altitude.
7. **physical object:** An object which is physical or tangible (and, therefore, does not itself appear in a digital archive). Examples of ‘physical objects’ include the planet Saturn and the Venus Express magnetometer. Note that an ASCII file describing Saturn is a digital object, not a physical object (nor a component of a physical object).
8. **resource:** The target (referent) of any Uniform Resource Identifier; the thing to which a URI points.

Note that the term **data element** is sometimes used as a synonym for **attribute** (or **class** or both). In PDS3 it was used frequently as a synonym for keyword; it is not used in this tutorial. The term **keyword** is avoided since **keyword** is more closely associated with search terms, as in publication keywords. The term **object**, a synonym for **class** in the PDS3 data model, is not used in that sense here; an **object** is *created from a class*.

D.4 PDS4 Data Dictionary Structure

Meta-attributes (attributes of attributes) are used to define attributes. For example, the attribute `axes` is defined using the meta-attributes **name** and **definition** (among others). **name** provides a common name for the attribute (“axes”) and **definition** provides a statement that describes the attribute (“a count of the axes”).

A subset of the ISO/IEC 11179 Metadata Registry reference model was chosen for the PDS4 Data Dictionary meta-attributes and structure. This standard ensures data system stability and interoperability.

The PDS4 Data Dictionary focuses on attributes and, to a lesser extent, on classes. A model of an attribute definition is shown in Figure 1. A PDS4 attribute is defined by the structure within the dashed line: the four orange boxes labeled “Attributes” (which contain *meta*-attributes) and the four blue boxes (containing decimal numbers) which represent classes. Definitions of the meta-attributes are given below. The **Bibliographic_Reference** class provides optional background information for the attribute definition, and the structure *outside* the dashed line is important for DD management; neither will be discussed further here.

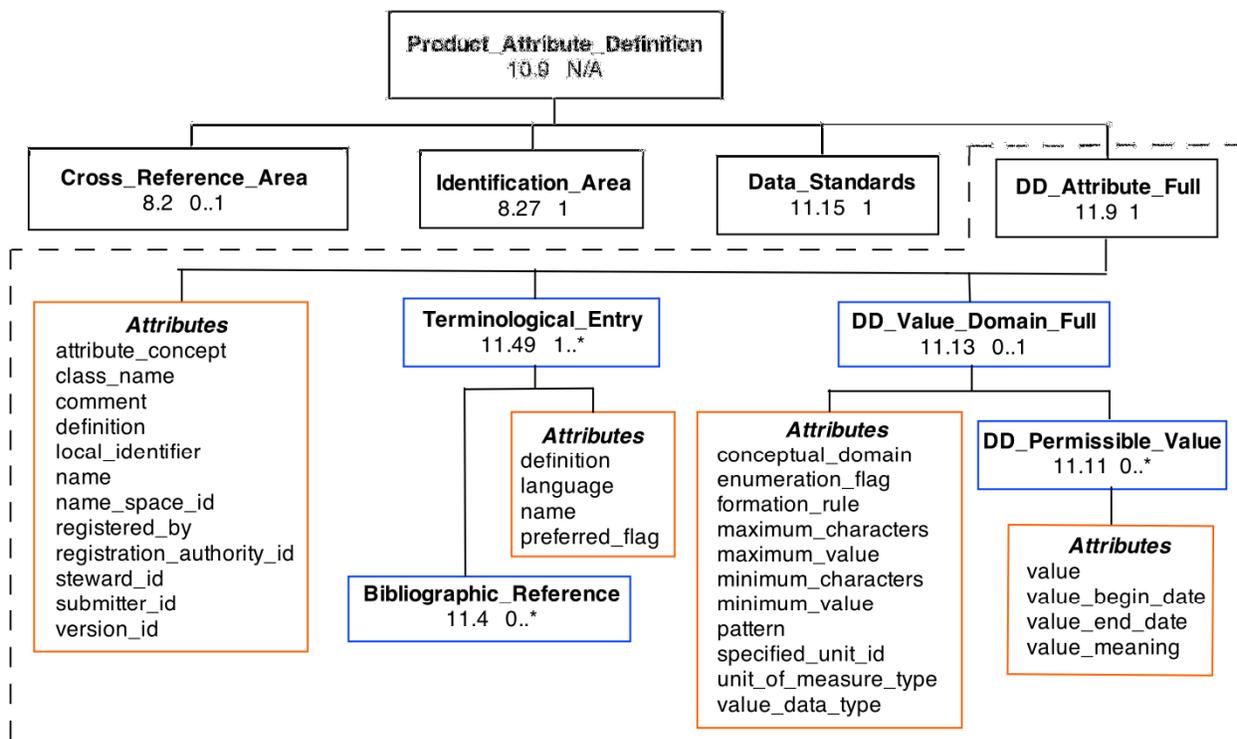


Figure 1. Model of a PDS4 Data Dictionary attribute definition. Boxes with decimal numbers represent classes; decimal numbers are the relevant section numbers in the PDS4 Information Model (version 0.5.0.0.g), and accompanying numbers show cardinality. Structure outside the dashed line is used for DD management.

Figures 2, 3, and 4 show examples of how attributes and their meta-attributes are presented through the PDS4 Data Dictionaries. Note that only some of the meta-attributes are displayed in the DD. Where a meta-attribute is shown as “required” but it does not appear in an example (Figures 2, 3, or 4), that is because the Data Dictionary does not display the attribute — often because the attribute has the same value throughout the Dictionary (for example 17 and 18 below). Note also that several terms are used as both meta-attributes here and attributes elsewhere (e.g., **name** and **comment**); their definitions are allowed to differ slightly depending on the context. The definitions here apply only to meta-attribute usage; see the Data Dictionary itself for definitions in the attribute context. Also, even though the ISO/IEC 11179 specification allows the inclusion of class definitions, this aspect of the model is not currently used; the PDS4 class definitions in the Data Dictionary are simply copied verbatim from the PDS4 Information Model.

• axes

```

steward_id: pds
name_space_id: pds
class_name: Array_2D
version_id: 0.5.0.0.g
attribute_concept: COUNT
definition: The axes attribute provides a count of the axes.
conceptual_domain: INTEGER
value_data_type: ASCII_Integer
minimum_value: 1
maximum_value: 16
permissible_value: 2
  
```

Figure 2 - PDS4 Unabridged Data Dictionary entry for the attribute 'axes' as used with class Array_2D

```
• sample_display_direction
  steward_id: pds
  name_space_id: pds
  class_name: Image_2D_Display
  version_id: 0.5.0.0.g
  definition: The sample_display_direction attribute is the preferred orientation of samples within a line
  for viewing on a display device. The default is right, meaning samples are viewed from left to right on
  the display. "sample_display_direction" must be used with "line_display_direction". Image rotation
  attributes such as TWIST_ANGLE, CELESTIAL_NORTH_CLOCK_ANGLE, and BODY_POLE_CLOCK_ANGLE
  are defined under the assumption that the image is displayed in its preferred orientation.
  value_data_type: ASCII_Short_String_Collapsed
  minimum_characters: 1
  maximum_characters: 255
  permissible_values:
    RIGHT
    DOW
    LEFT
    UP
```

Figure 3 - PDS4 Abridged Data Dictionary entry for the attribute 'sample_display_direction'

```
• exposure_duration
  steward_id: img
  name_space_id: img
  version_id: 0.3.1.1.Beta
  definition: This element provides the value of the time interval between the opening and closing of an
  instrument aperture (such as a camera shutter).
  value_data_type: ASCII_Real
  minimum_value: 0.0
  maximum_value: INF
  unit_of_measure_type: UnitOfMeasure_Time
  specified_unit_id: s
```

Figure 4 - PDS4 Abridged Data Dictionary entry for the attribute 'exposure_duration'

Meta-attributes are defined below.

1. The meta-attribute **attribute_concept** (required; supplied by the steward) provides the type of information (a classification) conveyed by the attribute. For example, `axes` has **attribute_concept** set to `COUNT` (Figure 2).
2. The meta-attribute **class_name** (required) provides the common name by which the class is identified; it is the class within which the attribute is used. Specification of the class-attribute pair is a unique entry within a data dictionary. **class_name** is set to `Array_2D` in Figure 2 and `Image_2D_Display` in Figure 3. Since the Abridged Data Dictionary collapses many class-attribute definitions into one, no value for **class_name** is shown in Figure 4.
3. The meta-attribute **comment** (optional) is a character string expressing one or more remarks or thoughts relevant to the attribute. No **comment** is shown in Figures 2, 3, or 4.

4. The meta-attribute **conceptual_domain** (optional; supplied by the steward) provides the domain to which the value has been assigned. In Figure 2 **conceptual_domain** is set to INTEGER; no value is shown for the Abridged DD entries in Figures 3 and 4.
5. The meta-attribute **definition** (required) provides a statement, picture in words, or account that defines the attribute. The **definition** used in the Data Dictionary is selected from one or more **definition** choices in **Terminological_Entry** using **preferred_flag**.
6. The meta-attribute **enumeration_flag** (optional) indicates whether there is an enumerated set of permissible values. If not specified, the default value 'F' (false) may be assumed. **enumeration_flag** is set to 'T' (true) for the examples in Figures 2 and 3; but it is not shown explicitly in the DD listings because presence of **DD_Permissible_Value** implies the value 'T' (see #16 below).
7. The meta-attribute **formation_rule** (optional) provides a user-friendly instruction for forming values of the attribute; **pattern** is a symbolic instruction, which is not intended to be user-friendly. **formation_rule** is not used in Figures 2, 3, and 4.
8. The meta-attribute **local_identifier** (required) is a character string which uniquely identifies DD_Attribute_Full within its label. In a complex label, **local_identifier** provides a shorthand term that can be used when making internal references within the label. No **local_identifier** is shown in Figures 2, 3, or 4.
9. The meta-attribute **maximum_characters** (optional) provides the upper, inclusive bound on the number of characters. **maximum_characters** is set to 255 in Figure 3, although 5 would have been sufficient for the **DD_Permissible_Values** listed.
10. The meta-attribute **maximum_value** (optional) provides the upper, inclusive bound on the value. **maximum_value** is set to 'INF' (infinity) in Figure 4.
11. The meta-attribute **minimum_characters** (optional) provides the lower, inclusive bound on the number of characters. **minimum_characters** is set to 1 in Figure 3.
12. The meta-attribute **minimum_value** (optional) provides the lower, inclusive bound on the value. **minimum_value** is set to 0.0 in Figure 4.
13. The meta-attribute **name** (required) provides a word or combination of words by which the attribute is known. The **name** used in the Data Dictionary is selected from one or more **name** choices in **Terminological_Entry** using **preferred_flag**. **name** is implicit in the first line of Figures 2, 3, and 4.
14. The meta-attribute **name_space_id** (required) is the abbreviation of the XML schema namespace container (see below) for this logical grouping of classes and attributes. The **name_space_id** is assigned by the steward (see below); the PDS steward has set **name_space_id** to 'pds' in Figures 2 and 3, and the IMG steward has set it to 'img' in Figure 4.
15. The meta-attribute **pattern** (optional) provides a symbolic instruction for forming values. **formation_rule** is a user-friendly instruction, which is intended to convey the same information in words that can be readily understood by a reader/user. **pattern** is not used in Figures 2, 3, and 4.
16. **DD_Permissible_Value** is a class, optionally used within attribute definitions; so it is not a meta-attribute *per se*, but its own attributes perform the same function. If **DD_Permissible_Value** appears,

it is accompanied by the attribute **enumeration_flag** set to 'T'. If multiple values are allowed, a separate **DD_Permissible_Value** definition is required for each. The meta-attributes under **DD_Permissible_Value** are:

- 16A. **value** (required for each **DD_Permissible_Value**) is a single, allowed numerical or character string value. The only permissible value for `axes` when used with `Array_2D` is 2 (Figure 2). In Figure 3 there are four possible values for `sample_display_direction`: "RIGHT", "DOWN", "LEFT", and "UP". Note that **enumeration_flag** is 'T' in Figures 2 and 3. In Figure 4 there is no enumeration; an infinite number of values is possible.
- 16B. **value_begin_date** (optional) provides the first date on which the permissible **value** is in effect. **value_begin_date** is not used in Figures 2, 3, and 4.
- 16C. **value_end_date** (optional) provides the last date on which the permissible **value** is in effect. **value_end_date** is not used in Figures 2, 3, and 4.
- 16D. **value_meaning** (required for each **DD_Permissible_Value**) is the meaning or semantic content of the associated permissible **value**. Currently there are no examples of **value_meaning** in the Data Dictionary; however, the "PDS4 Data Type Definitions" chapter in the DD achieves the same result (but only for **data_type** values). **value_meaning** is not used in Figures 2, 3, and 4.
- 17. The meta-attribute **registered_by** (required) provides the name of the person or organization that registered the attribute. **registered_by** is not used in Figures 2, 3, and 4.
- 18. The meta-attribute **registration_authority_id** (required) is the name of the organization that registered the attribute definition. For all PDS attributes, **registration_authority_id** will be set to '0001_NASA_PDS_1'. **registration_authority_id** is not shown in Figures 2, 3, and 4.
- 19. The meta-attribute **specified_unit_id** (optional) provides the units in which other meta-attributes such as **maximum_value** and **DD_Permissible_Value** are given. **specified_unit_id** is 's' in Figure 4.
- 20. The meta-attribute **steward_id** (required) is the person or organization who manages the set of registered attributes and classes. `sample_display_direction` and `axes` have both been assigned to 'pds' (Figures 2 and 3); `exposure_duration` has been assigned to 'img' (Figure 4).
- 21. The meta-attribute **submitter_id** (required) is the name of the author who submits the attribute definition to the steward. **submitter_id** is not used in Figures 2, 3, or 4.
- 22. **Terminological_Entry** is a required class; there may be multiple instances of **Terminological_Entry** within a single **DD_Attribute_Full**. Meta-attributes under **Terminological_Entry** are:
 - 22A. **definition** (required) provides a statement, picture in words, or account that defines the term. **definition** of `axes` is "a count of the axes" (Figure 2).
 - 22B. **language** (required) indicates the language used for the **name** and **definition** of the term. **language** is not shown in Figures 2, 3, and 4.
 - 22C. **name** (required) provides a name for the term in the **language** selected.
 - 22D. **preferred_flag** (required) indicates whether this entry is preferred over all other entries — for

example, because of a **language** choice.

Terminological_Entry is not shown explicitly in Figures 2, 3 ,or 4.

23. The meta-attribute **unit_of_measure_type** (optional) provides the named grouping of units to be used for this attribute. **unit_of_measure_type** is set to `UnitOfMeasure_Time` in Figure 4.
24. The meta-attribute **value_data_type** (required) provides the data type used to represent the value. The **value_data_type** for `axes` is `ASCII_Integer` (Figure 2), meaning that values like -255, 0, and 7 are acceptable so far as storage is concerned (that -255 is not a meaningful count of axes is addressed using **DD_Permissible_Value**). The **value_data_type** for `sample_display_direction` is `ASCII_Short_String_Collapsed` (Figure 3), meaning that special characters such as ‘line-feed’ and ‘tab’ are replaced by spaces and then each multi-space sub-string is reduced to a single space. **value_data_type** is an enumerated attribute; acceptable values can be found in the PDS4 Data Dictionary.
25. The meta-attribute **version_id** (required; supplied by steward) identifies the version of the attribute’s definition. **version_id** is set to `0.5.0.0.g` in Figures 2 and 3 and is set to `0.3.1.1.Beta` in Figure 4.

D.5 Management of Attributes and Classes

Management of PDS4 includes assignment of responsibility for maintenance of each attribute and class in the Data Dictionary. The ISO/IEC 11179 reference model provides two attributes for this purpose: ‘registration authority’ and ‘steward’. ‘Namespace’ also plays a role but only for the XML implementation.

A **registration authority** is an organization responsible for maintaining a registry. The ISO/IEC 11179 reference model allows many registration authorities, each of which is uniquely identified. Each registration authority has, by definition, its own model and therefore, implicitly, its own local dictionary. Each registration authority can design, develop, and manage its own model and dictionary using any data modeling methodology and independently of the other Registration Authorities.

The registration authority for the Planetary Data System is ‘0001_NASA_PDS_1’. PDS has designed, developed, and managed its model using the ‘object oriented’ methodology. An important constraint levied by the model is that each attribute and class must have a unique name within the model. But PDS attributes and classes may duplicate those maintained by other registration authorities.

A **steward** is a person or organization who manages a set of registered attributes and classes, typically as an agent of another or others. Each attribute and class in the PDS4 Data Dictionary is assigned to a single steward. Stewards for PDS4 include PDS, the discipline nodes, and any mission wishing to conform to the PDS4 Information Model. A registration authority must have at least one steward; but it may have many. Stewards are uniquely identified within a registration authority. A single steward may operate across many registration authorities.

Namespace is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols (*i.e.*, names). An identifier defined in a namespace is associated with that namespace. The same identifier may be independently defined in multiple namespaces. Namespaces are not a functional component of the PDS4 Information Model or Data Dictionary; rather, they are assigned and used for implementation into XML Schema. A steward may ask for a namespace from the PDS Namespace Registry

Service, which will make the assignment if the namespace has not previously been assigned.

D.6 Acknowledgements

The PDS4 Data Dictionary and the PDS4 Information Model are joint efforts involving representatives from each of the PDS nodes functioning as the PDS4 Data Design Working Group.