PDS4 Concepts

Data Design Working Group
3 September 2015
Version 1.4.0

Contents

PDS4 Concepts 1
Contents 2
Change Log 5
Version 1.4.0 5
I. Introduction 7
1. Background 7
1.1 The Need for Standards 7
1.2 The Archiving Process 7
2. This Document 8
2.1 Outline 8
2.2 Audience 8
2.3 Related Documentation 8
II. Overview of PDS4 9
3. Data Structures 9
3.1 Observational Data 9
3.2 Documents 9
3.3 Supplementary Information 10
4. XML and XML Schema 10
4.1 What is XML? 10
4.2 Why XML? 11
4.3 What is XML Schema? 11
4.4 Why XML Schema? 11
4.5 Terminology 12
5 Namespace 13
6. Data Dictionaries 14
6.1 What is a Data Dictionary? 14
6.2 PDS4 Data Dictionaries 14
6.3 Creating Data Dictionaries 15
6.4 Using Data Dictionaries 15

7. Labels 16
7.1 Organization 16
7.2 Creation 17
7.3 Validation 17
8. Archive Organization 17
8.1 Products 18
8.2 Collections 20
8.3 Members of Collections 20
8.4 Bundles 21
8.5 Logical Identifiers 21
8.6 Version Identifiers 22
III. PDS4 Data Structures 23
9. Scalars 23
9.1 Character Types 23
9.2 Binary Types 23
10. Arrays 24
10.1 Elements 24
10.2 Axes 24
10.3 Storage Order 24
10.4 Additional Considerations 25
11. Tables 25
11.1 Character Tables 25
11.2 Binary Tables 26
11.3 Records and Fields 27
12. Parsable Byte Streams 27
12.1 Flat Text 27
12.2 Flat UTF-8 Text 28
12.3 Delimited Tables 28
13 Encoded Byte Stream 29
13.1 PDF/A 29
13.2 Other Formats 29
Appendix A – PDS4 Glossary 31

Appendix B – PDS4 Acronyms and Abbreviations 37
Appendix C — PDS4 Data Dictionary 38
C.1 Introduction and Purpose 38
C.2 Related Documents 38
C.3 Terminology 38
C.4 PDS4 Data Dictionary Structure 39
C.5 Management of Attributes and Classes 44

Change Log

Version 1.4.0

Date	Section(s)	Changes
2013-05-01	All	Original (v1.0.0) by Anne Raugh, Ron Joyner, and Dick Simpson
2015-07-08	Various	Changed document version and date
2015-04-30	Change Log	Added Change Log
2015-05-01	Figures C-1 through C-4	Updated figures and captions for Information Model 1.4.0.0 and current Abridged Data Dictionary format.
2015-05-09	C.4	Added “in a data dictionary” to first sentence. Changed “four blue boxes” to “five blue boxes” in third paragraph.
2015-05-10	C.4 meta-attribute definitions	Added “8. Instance_id” under Terminological_Entry. “type” and “unit_id” were removed from under unit_of_measure_type; specified_unit_id was moved from under unit_of_measure_type to being directly under DD_Value_Domain_Full. Meta-attribute definitions were renumbered as needed. Text was updated to be consistent with IM 1.4.0.0 and the figures.
2015-05-10	6.3	Removed everything after the first paragraph and added a new paragraph that reflects the current approach to local data dictionary creation.
2015-05-10	7	Added close quote at end of paragraph.
2015-05-10	7.3	Made “’tailored’ schemas” plural in second line, since the consulting node may provide more than one.
2015-05-10	8	Reworded the last sentence.
2015-05-10	9.1	Reworded the final sentence, correcting the delimiter pair. Change “TAB” to “Tab” in the next to last sentence.
2015-05-10	10.	Removed the final sentence, since N=4 and larger arrays are now being used in PDS4 products
2015-05-10	6.4, 7, 10.1, 10.2	Replaced “global” by “common” when it refers to the PDS4 data dictionary used by everyone.
2015-05-10	10.3	In the second paragraph replaced “notationally, the leftmost axis” by “notationally, the leftmost subscript”.
2015-05-10	A (secondary member)	Changed “… Voyager images were in one primary collection …” by “… Voyager images were primary members of one collection …”
2015-05-10	Table of Contents	Updated.
2015-05-27	Figure C-4 caption	Corrected ‘exposure_duration’ to ‘altitude in class Telescope’.
2015-07-08	1	Minor wording changes to this ‘Background’ paragraph.
2015-07-08	9	Added a footnote to the first sentence clarifying that ‘scalar’ is not a *storage* structure.
2015-07-08	11	Expanded the footnote to say explicitly that Table_Delimited is a parsable byte stream.
2015-07-08	12.1	Dropped the final sentence, since it added nothing.

2015-07-08	C.4	Meta-attribute #9 and meta-attribute #11: changed minimum_characters and maximum_characters to 'None' for consistency with Figure C-3. However, the values in C-3 are wrong, so this needs to be corrected again if/when CCB-118 is approved and implemented.
2015-07-08	C.4	Meta-attribute 17A: Noted that enumeration_flag, which was shown explicitly in earlier Data Dictionaries, in now hidden and must be inferred.
2015-09-03	Various	CCB-124: Late typo and comparable changes requested by Tom Stein in his JIRA comments of 2015-08-13. Style consistency awaits adoption of style guidelines for PDS documents. See detailed responses in Simpson JIRA comments of 2015-09-03.

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 references are [1] and [4].

1. Background

NASA's Planetary Data System (PDS) is a federation of 'discipline nodes,' each specializing in a subset of planetary science such as geosciences, atmospheres, and rings. PDS has been operational for more than two decades, most of that time in its version 3 (PDS3). A lot has changed since PDS3 was introduced; 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[1], 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 should be included — calibrations, documentation, etc. When data production starts, the consulting node provides validation

[1] The terms 'data provider' and 'data preparer' are used interchangeably throughout this document to mean the person or organization archiving data with PDS.

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 (including vocabulary) 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 manuals for PDS4 software tools.

2.3 Related Documentation

[1] *PDS4 Standards Reference*, see http://pds.nasa.gov/pds4/doc/index.shtml for current and past versions.
[2] *PDS4 Data Provider's Handbook*, see http://pds.nasa.gov/pds4/doc/index.shtml for current and past versions.
[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] *PDS4 Information Model*, see http://pds.nasa.gov/pds4/doc/index.shtml for current and past versions.

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. If data cannot be described in terms of these PDS4 'base' structures, they cannot be archived under PDS4 and must be revised. Note that not all PDS3 structures are compatible with PDS4.

3.1 Observational Data[2]

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 [1] 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/A[3] standard as the archival format for complex documents. UTF-8[4]

---

[2] 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.'

[3] "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 delimited tables 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 into appropriate contexts 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

Some supplementary information, such as instrument descriptions, is 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.

Supplementary information must be in the same four 'base' structures as were described above.

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.

[4] 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 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 tags[5] used in the XML document. XML defines the syntax for the tags, but not their names or meanings. PDS4 uses XML Schema[6] 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

---

[5] 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 Appendix A.

[6] *Schema* is a Greek word meaning *shape* or *plan*; its plural is *schemata*. In English both *schemata* and *schemas* are used [see www.wikipedia.org]; 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 of 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 tag in the file. The associated XML element contains all the other elements that comprise the document. PDS labels are well-formed XML documents.

An *XML attribute* is a qualifier for 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 content of `<movie:title>` will no doubt be a string, but the content 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.[7]

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 in 1634 with forging a common French language; 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.

---

[7] 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. When used in the context of 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.

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.

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; node and data preparer data dictionaries 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

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.

Creating a local data dictionary is not an exercise to be undertaken lightly. It should be conducted under supervision of a consulting and/or lead node to ensure conformance with PDS4 requirements for LDDs. Typically the work begins with a semantically and syntactically valid local data dictionary template, which is then completed for the classes and attributes to be added, restricted, or extended in the local dictionary. Appendix C provides background, which may be useful in understanding the scope of the work.

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 common 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.

7. Labels

With the PDS common data dictionary (and, possibly, node and mission dictionaries), it is possible to construct labels for the products in an 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. The combination of a PDS label plus what it describes is called a 'product'.

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)

Immediately after the XML header (see lines 1-2 in the example below) is a root tag, `<Product_Observational>`, which, paired with `</Product_Observational>`, envelopes everything else in the label. Inside this complex XML element (only a few details are shown in the example), information is organized into the 'areas' described above (except Reference List Area, which is optional), which are complex XML elements themselves. The hierarchy continues downward to the lowest level — simple XML

elements which specify values . 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>
          ...
        <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. The schema approach requires that code be written and then executed to generate labels. A template looks exactly like the finished label except that dummy variables have been inserted where values change; a 'search and replace' procedure can then be used to fill in the correct values when a label must be generated.

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' schemas 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. PDS personnel have developed a stand-alone validation tool which uses schemas and Schematron files to search for and validate specific patterns in labels. 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 of 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 to ensure proper storage and retrieval. This section describes the hierarchy used to organize archives created by data preparers both physically and logically. Whereas Figure 8-1 shows the organization from the top down, the sections below 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. The header is a second digital object within the data file; we need to describe and track both the header object and the image (or table) object. 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'.

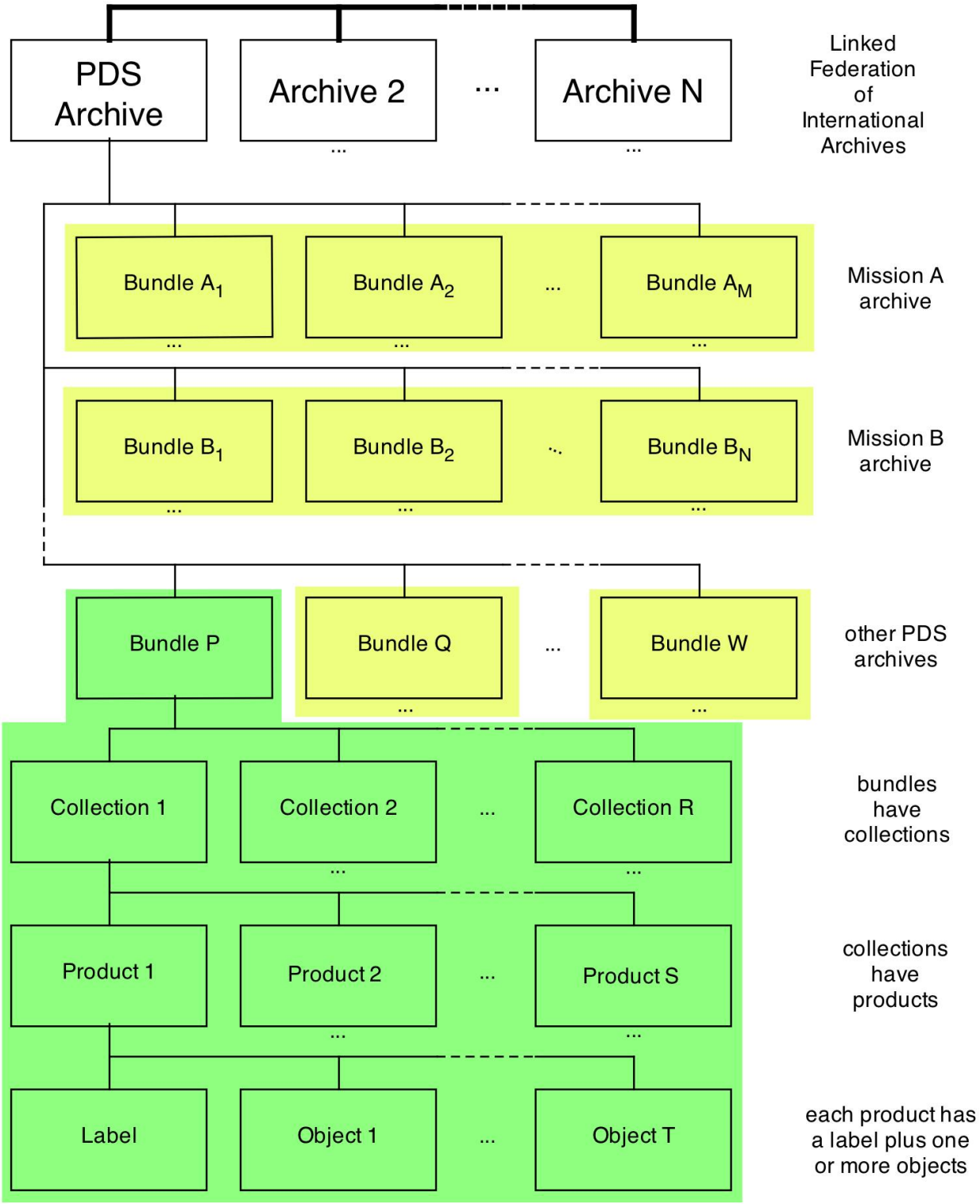

Figure 8-1. Organization of archives, including links from PDS to other archive systems, which interact by exchanging XML documents. Some missions (A and B) may create multiple bundles. A single bundle may have several collections, each of which has several products comprising a label and one or more objects (green). Other bundles (yellow) would have similar structures.

Less commonly, the digital objects, which need to be tracked together as a single product, are spread over multiple files. For example, while red, green, and blue arrays that form a color image could be stored in single file and labeled and tracked as a single product, they could also be in three separate files.

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 product 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, 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.

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 could 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.

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 prepending 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`.

8.6 Version Identifiers

Products, collections, and bundles may evolve over time as calibration or other processing changes lead to improved versions. PDS appends a version identifier (VID) of the form `M.n` to logical identifiers to indicate different versions. The suffix is separated from the LID by a double colon — for example

```
urn:nasa:pds:sunburn-1:solcam-cr1-raw:image01321::1.0
```

would be the first version of product `image01321`. There are some circumstances in which the versioned logical identifier (LIDVID) is more appropriate and others in which the LID may be preferred. When a product is requested and only the LID is given, PDS will default to the most recent version. Products which have undergone distinctly different processing (as opposed to better calibration), should be given a new LID.

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. 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.

9. Scalars

Scalars are the smallest storage units in the PDS4 system[8]. 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.

9.1 Character Types

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 requires the `Carriage-Return Line-Feed` pair as the record delimiter in both `Table_Delimited` and `Table_Character`.

9.2 Binary Types

Binary data types must be used in arrays and in binary tables. Acceptable binary data types include signed and unsigned integers of various bit lengths, IEEE 754 floating point representations of 4- and 8-byte lengths, and complex formats constructed from floating point components. Two byte orders are recognized for binary numbers — least significant byte first (LSB) and most significant byte first (MSB).

---

[8] However, 'scalar' is not a PDS4 storage *structure*; scalars are part of larger structures.

10. Arrays

The simplest structure available for data storage is the (homogeneous) array — that is, an N-dimensional structure in which all the elements have the same data type. This is the structure underlying all images, spectral cubes, maps, and similar structures. Defining an array requires definition of the element type, and then definition of the number and lengths of each of the axes.

10.1 Elements

Any of the binary types based on 8-bit bytes listed in section 9.2 may be used as the data type for an array element. Array elements may be associated with a number of characteristics such as units of measure, offsets, and scaling factors. The data preparer may also designate specific 'flag' values to be used — for example, to indicate saturated pixels in an image. In a PDS4 XML label, the element and its attributes are defined using the `Element_Array` class in the common PDS data dictionary.

10.2 Axes

Each array will have one or more axes. In a PDS4 label, each axis is defined using the `Axis_Array` class in the common 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 a PDS4 label, axis order is stated explicitly using a sequence number in each axis definition. If the axes of an array correspond to subscripts, then the axis with sequence number 1 would be the first (leftmost) subscript, axis number 2 the next, and so on.

10.3 Storage Order

The individual bytes 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 slowly; that along axis 2 next most slowly; 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 row-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 used by the C programming language, and it is the default storage order for 2-dimensional images under the PDS3 standard.

10.4 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 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.

11. Tables

Tables — more specifically fixed-width tables[9] — 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 not contain binary fields.

11.1 Character Tables

Each field in a character table must have one of the allowed character data types (section 9.1. Each record in a character table must end with a two-byte record delimiter consisting of the ASCII carriage return character and the ASCII line feed character.

Character tables often 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 (printable 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 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. The summed widths of all defined fields subtracted from the total record length should be the number of gutter space bytes plus the record delimiters.

---

[9] Fixed-width tables are distinguished from tables with records of variable length — the `Table_Delimited` class, which is actually a parsable byte stream (see Section 12.3)

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

- 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 with 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 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.

11.2 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 mentioned in Section 9.2, though some of the character types mentioned in Section 9.1 may also appear. Dates and times, simple character strings, and flags (for example, '0' and '1') are among the most common character fields in binary tables.

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

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

- 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.

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. However, 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.

11.3 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 attributes. 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.

12. 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 also recognizes SPICE kernels, certain header structures, and XML files as falling within the definition of a parsable byte stream.

12.1 Flat Text

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

- `offset` — displacement in bytes of the start of the text from the beginning of the parent structure (often set to "0")
- `record_delimiter` — set to the `carriage-return line-feed` pair

- `encoding_type` — set to `CHARACTER`, which allows UTF-8 and its subset ASCII
- `parsing_standard_id` — the non-PDS standard which makes the stream parsable

Note that record delimiters (as mentioned for the example above) are only required if mandated by the external standard.

12.2 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 12.1) 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.

12.3 Delimited Tables

In contrast to fixed-width tables (Section 11), 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 programs[10].

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` pairs).

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>
```

13 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.

13.1 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 (possibly listed as "PDF/A-1a" and "PDF/A-1b" in some format menus).

13.2 Other Formats

[10] 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 – 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.

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. Product_Bundle, the bundle's manifestation, is itself a product (because it is simply a list embedded within a label); but it is not a *basic* product. 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. The collection's manifestation, Product_Collection, is itself a product (because it is simply a list, with its label); but it is not a *basic* product.

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).

consulting node: A PDS discipline node assigned as the contact for a mission, instrument, or project.

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

data dictionary: A repository for definitions of classes and attributes.

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

data preparer: Same as data provider.

data provider: A person or organization that assembles archival data for delivery to PDS.

data structure: A particular way of storing data in a computer that facilitates efficient use.

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.

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.

discipline area: That part of a label which is specified by a discipline.

encoded byte stream: A byte stream that may only be interpreted after it has been 'decoded' according to some well known standard.

entity: Something that has a distinct, separate existence.

extension: (1) See subclass. (2) The character string following the last period in a file name.

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: 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.

lead node: One of several consulting nodes designated as the PDS coordinator and primary contact with a mission.

local: (1) Within a single label. (2) Within an archiving entity — *e.g.*, local data dictionary.

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.

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 attribute *type* within class Investigation_Area).

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.

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.

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 attribute *type* within class Investigation_Area).

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

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.

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).

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. A product's member status (primary or secondary) is based on its first association with the collection. Although the product may be omitted from a later version of the collection, it retains its primary or secondary member status through all subsequent versions of the collection based on its initial association. In a similar way, collections are categorized as having either primary or secondary 'member status' in their bundles.

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.

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'.

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: Data which have been recovered from storage and successfully prepared for archive in PDS.

restriction: A limit placed on the range of a variable; specifically, the narrowing of possible choices for a class or attribute. For example, attribute `axes` may have values between 1 and 16 in the definition of Array, but it is restricted to the value '2' in Array_2D.

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

science bundle: Observational data from a science investigation, documentation, and other supplementary data organized into a bundle structure for delivery to PDS.

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. A product's member status (primary or secondary) is based on its first association with the collection. Although the product may be omitted from a later version of the collection, it retains its primary or secondary member status through all subsequent versions of the collection based on its initial association. In a similar way, collections are categorized as having either primary or secondary 'member status' in their bundles.

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: In PDS4 a subclass is a class extension. 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.

supplementary data: 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 'carriage-return line-feed' record delimiter.

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

tagged digital object: A digital object paired with its companion description object. [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. [Note: In the OAIS RM this pair is known as an 'information object'].

version identifier (**VID**): An identifier which identifies the version of something else.

versioned identifier (**LIDVID**): The concatenation of a logical identifier (LID) with a version identifier (VID).

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

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

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 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 Information Model, which is propagated to the PDS4 Data Dictionary.

XML label: A label written using XML.

XML root tag: The first (and highest-level) XML tag 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.

Appendix B – PDS4 Acronyms and Abbreviations

ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
COSPAR	Committee on Space Research
DD	Data Dictionary
DDB	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 C — PDS4 Data Dictionary

C.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 appendix 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.

C.2 Related Documents

- Controlling Documents
  - PDS4 Information Model Specification — The source for PDS4 class, attribute, and data type definitions. See http://pds.nasa.gov/pds4/doc/index.shtml for current and past versions.
  - ISO/IEC 11179:3 Registry Metamodel and Basic Attributes Specification, 2003 - The reference schema for the PDS4 data dictionary.
- Reference Documents
  - PDS4 Glossary - The source for terms used across the Planetary Data System in its version 4 (PDS4) (Appendix A in the *PDS4 Concepts* document; see http://pds.nasa.gov/pds4/doc/index.shtml for current and past versions).
  - 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.

C.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 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, ...

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 (the table) and another (a 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 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).

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 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).

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**.

C.4 PDS4 Data Dictionary Structure

Meta-attributes (attributes of attributes) are used to define attributes in a data dictionary. 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 C-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 five blue boxes (containing decimal numbers) which represent classes. Definitions of the meta-attributes are given below. The **External_Reference_Extended** class provides optional background information for the attribute definition, and the structure *outside* the dashed line is important for Data Dictionary management; neither will be discussed further here.

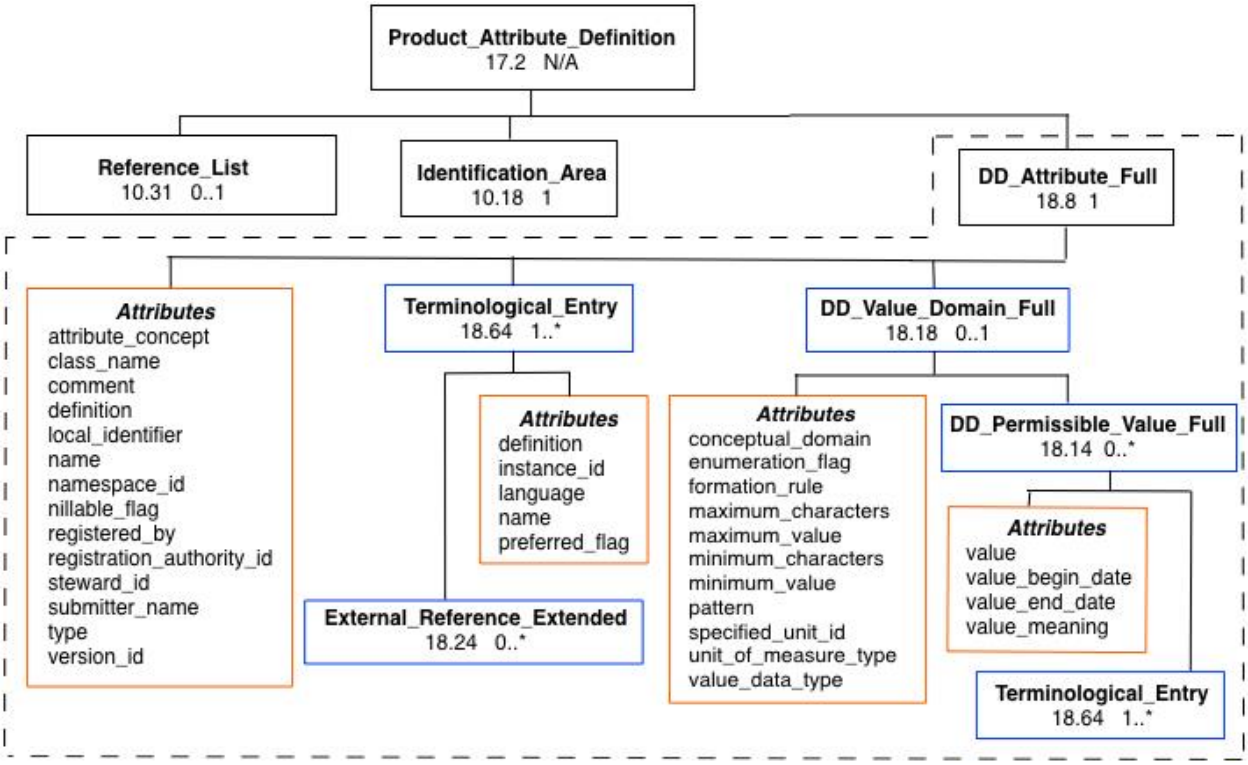

Figure C-1. Model of a PDS4 Data Dictionary attribute definition. Boxes with decimal numbers represent classes; decimal numbers are the relevant section numbers in the PDS4 Information Model (version 1.4.0.0), and accompanying numbers show cardinality. Structure outside the dashed line is used for Data Dictionary management. Detailed substructure under both Terminological_Entry boxes is the same; in the lower right it is not shown because of space limitations.

Figures C-2, C-3, and C-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 Data Dictionary. Where a meta-attribute is required but does not appear in an example (Figures C-2, C-3, or C-4), that is because the Data Dictionary does not display the attribute — often because the attribute has the same value throughout the Dictionary. 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 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.

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 C-2).

2. The meta-attribute **class_name** (required) provides the common name by which the parent 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 C-2, 'Display_2D_Image' in Figure C-3, and 'Telescope' in Figure C-4.

axes in Array_2D

***Name:* axes**			***Version Id:* 1.0.0.0**
Description: The axes attribute provides a count of the axes.			
Namespace Id: pds	*Steward:* pds	*Class Name:* Array_2D	*Type:* ASCII_Integer
Minimum Value: 1	*Maximum Value:* 16	*Minimum Characters:* None	*Maximum Characters:* None
Unit of Measure Type: None	*Default Unit Id:* None	*Attribute Concept:* Count	*Conceptual Domain:* Integer
Status: Active	*Nillable:* false	*Pattern:* None	
Permissible Value(s)	*Value*	*Value Meaning*	
	2	Array_2D has 2 axes	

Figure C-2. PDS4 Abridged Data Dictionary entry for the attribute `'axes'` as used with class Array_2D.

sample_display_direction in Display_2D_ Image *Deprecated*

***Name:* sample_display_direction**			***Version Id:* 1.0.0.0**
Description: The sample_display_direction attribute provides the preferred orientation of samples within a line for viewing on a display device. The attribute sample_display_direction must be used with line_display_direction.			
Namespace Id: pds	*Steward:* pds	*Class Name:* Display_2D_Image *Deprecated*	*Type:* ASCII_Short_ String_Collapsed
Minimum Value: None	*Maximum Value:* None	*Minimum Characters:* None	*Maximum Characters:* None
Unit of Measure Type: None	*Default Unit Id:* None	*Attribute Concept:* Direction	*Conceptual Domain:* Short_String
Status: Active	*Nillable:* false	*Pattern:* None	
Permissible Value(s)	*Value*	*Value Meaning*	
	Right	The preferred orientation of samples within an line for viewing on a display device is Right	

Figure C-3. PDS4 Abridged Data Dictionary entry for the attribute `'sample_display_direction'`. Note that both the attribute and its parent class are no longer recommended for new uses (deprecated).

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 C-2, C-3, or C-4.

4. The meta-attribute **conceptual_domain** (optional; supplied by the steward) provides the domain to which the value has been assigned. For `axes`, **conceptual_domain** is set to INTEGER (Figure C-2).

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.** The selected **definition** is shown in Figures C-2, C-3, and C-4 in the field titled *Description*.

altitude in Telescope

***Name:* altitude**			***Version Id:* 1.0.0.0**
Description: The altitude attribute provides the height of anything above a given reference plane.			
Namespace Id: pds	*Steward:* pds	*Class Name:* Telescope	*Type:* ASCII_Real
Minimum Value: None	*Maximum Value:* None	*Minimum Characters:* None	*Maximum Characters:* None
Unit of Measure Type: Units_of_Length	*Default Unit Id:* m	*Attribute Concept:* Number	*Conceptual Domain:* Real
Status: Active	*Nillable:* false	*Pattern:* None	
Permissible Value(s)	*No Values*		

Figure C-4. PDS4 Abridged Data Dictionary entry for the attribute `'altitude'` in class Telescope.

6. The meta-attribute **enumeration_flag** (optional) indicates whether there is an enumerated set of permissible values. If not specified, the default value 'false' may be assumed. **enumeration_flag** is set to 'true' for the examples in Figures C-2 and C-3; this can be inferred from the fact that *Permissible Values* are listed. In Figure C-4 **enumeration_flag** can be inferred to be 'false'.

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 C-2, C-3, and C-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 C-2, C- 3, or C-4.

9. The meta-attribute **maximum_characters** (optional) provides the upper, inclusive bound on the number of characters. **maximum_characters** is set to 'None' in Figure C-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 '16' in Figure C-2.

11. The meta-attribute **minimum_characters** (optional) provides the lower, inclusive bound on the number of characters. **minimum_characters** is set to 'None' in Figure C-3.

12. The meta-attribute **minimum_value** (optional) provides the lower, inclusive bound on the value. **minimum_value** is set to '1' in Figure C-2.

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 shown in Figures C-2, C-3, and C-4 where it is 'axes', 'sample_display_direction', and 'altitude', respectively.

14. The meta-attribute **namespace_id** (required) is the abbreviation of the XML schema namespace (see below) for this logical grouping of classes and attributes. The **namespace_id** is assigned by the steward

(see below); the PDS steward has set **namespace_id** to 'pds' in Figures C-2 and C-3, and C-4.

15. The meta-attribute **nillable_flag** (required) indicates whether the attribute is allowed to take on the value 'nil' — that is, to have no value at all. The value is set to 'false' in all three examples.

16. 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 set to "None" in Figures C-2, C-3, and C-4.

17. 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 'true'. If multiple values are allowed, a separate **DD_Permissible_Value** definition is required for each. The meta-attributes under **DD_Permissible_Value** are:

    17A. **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 C-2). In Figure C-3 there is only one possible value for `sample_display_direction`: 'right'. Note that **enumeration_flag** is (implicitly) 'true' in Figures C-2 and C-3. In Figure C-4 there is no enumeration; the (hidden) **enumeration_flag** is 'false' and an infinite number of values is possible.

    17B. **value_begin_date** (optional) provides the first date on which the permissible **value** is in effect. **value_begin_date** is not used in Figures C-2, C-3, and C-4.

    17C. **value_end_date** (optional) provides the last date on which the permissible **value** is in effect. **value_end_date** is not used in Figures C-2, C-3, and C-4.

    17D. **value_meaning** (required for each **DD_Permissible_Value**) is the meaning or semantic content of the associated permissible **value**. Definitions are to the right of values in Figures C-2 and C-3.

18. The meta-attribute **registered_by** (required) provides the name of the person or organization that registered the attribute. **registered_by** is not shown in Figures C-2, C-3, and C-4.

19. 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** is set to '0001_NASA_PDS_1'. **registration_authority_id** is not shown in Figures C-2, C-3, and C-4.

20. The meta-attribute **specified_unit_id** (optional) gives the units in which other meta-attributes (**maximum_value** , **minimum_value**, and **DD_Permissible_Value)** are given. **specified_unit_id** is shown as *Default Unit Id* in Figure C-4 and is given as 'm'.

21. The meta-attribute **steward_id** (required) is the person or organization that manages the set of registered attributes and classes. **steward_id** has been set to 'pds' in Figures C-2, C-3, and C-4.

22. The meta-attribute **submitter_name** (required) is the name of the author who submitted the attribute definition to the steward. **submitter_id** is not shown in Figures C-2, C-3, and C-4.

23. **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:

23A. **definition** (required) provides a statement, picture in words, or account that defines the term. **definition** of `axes` is "a count of the axes" (Figure C-2).

23B. **instance_id** (optional) is the formal name of the Terminological_Entry.

23C. **language** (required) indicates the language used for the **name** and **definition** of the term. **language** is not shown in Figures C-2, C-3, and C-4.

23D. **name** (required) provides a name for the term in the **language** selected. The selected name appears as **name** under DD_Attribute_Full (#13 above).

23E. **preferred_flag** (required) indicates whether this entry is preferred over all other entries — for example, because of a **language** choice. If preferred, the 'definition' here becomes the definition under DD_Attribute_Full (#5 above) and the 'name' here becomes the name under DD_Attribute_Full (#13 above).

Terminological_Entry is not shown explicitly in Figures C-2, C-3, and C-4.

24. The meta-attribute **type** (required) identifies the attribute as coming from PDS3 or PDS4. It is not shown in the examples.

25**.** The meta-attribute **unit_of_measure_type** (optional) provides the named grouping of units to be used for this attribute. It names a class, of which there are about two dozen choices. **unit_of_measure_type** is set to 'Units_of_Length' in Figure C-4.

26.The meta-attribute **value_data_type** (required) provides the data type used to represent the value and is shown as *Type* in the examples. The **value_data_type** for `axes` is ASCII_Integer (Figure C-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 C-3). The **value_data_type** for `altitude` is 'ASCII_Real'. **value_data_type** is an enumerated meta-attribute; acceptable values can be found in the PDS4 Data Dictionary.

27. The meta-attribute **version_id** (required; supplied by steward) identifies the version of the attribute's definition. **version_id** is set to '1.000' in Figures C-2, C-3, and C-4.

C.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 identifier 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.