Data Providers' Handbook

Archiving Guide to the PDS4 Data Standards

DRAFT

PDS

Planetary Data System

National Aeronautics and Space Administration

Data Design Working Group

October 2010

Version 0.22.3

CHANGE LOG

Revision	Date	Description	Author
0.1	Mar 30, 2009	Initial draft based on information collected by the Data System Working Group.	R.Joyner
0.2	Aug 6, 2009	Updated versions of all Classes	R. Joyner
0.2.1	2010-08-31	Complete overhaul, but only partly successful through page 25	R. Simpson
0.22	Aug 31, 2010	Integrate Simpson and Joyner docs	R. Joyner etal
0.22.1	2010-09-22	Edits through Chapter 3, except Chapter 2	Simpson
0.22.2	2010-09-24	Added comments from Mitch Gordson	Simpson
0.22.2	2010-10-01	Significant edits to Chapter 4	Simpson
0.22.3	2010-10-25	Significant edits to Chapters 1,3-6	Simpson

LIEN LOG

Revision	Date	Description
0.22	Sept 13, 2010	1. XML labels in Section 7 need to be updated 2. Organization of sections needs help 3. Parsable Byte Stream vs Unencoded Stream Base 4. Collection_Archive vs Archive_Collection 5. Bundle_Archive vs Archive_Bundle 6. Example logical_IDs need to conform to rules 7. Examples need to be loaded into repository and find permanent location for repository 8. Table of Association_types needs work 9. Should be some mention of APG (workflow) 10. Need to add section on context products and where to locate 11. Add section on Node_Area and Mission_Area
0.22.1-0.22.2	2010-09-22/24	Anything of the form [Note for later: …]

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose

The Data Providers Handbook (DPH) is a guide for preparation of data being submitted to the Planetary Data System (PDS). The document should be used in conjunction with the PDS Standards Reference (PDSSR) [2] and the Planetary Science Data Dictionary (PSDD) [3]. All three documents have been updated for version 4 of the PDS (PDS4)[1].

While, the PDS4 Standards Reference remains the definitive source for ensuring data meet the PDS4 archive criteria, the DPH functions more in the capacity of a tutor/coach to provide information and examples to guide data providers in the design and preparation of data to be archived with the PDS

1.2 Audience

The DPH is written for scientists and engineers in the planetary science community who are planning to submit new or restored data to PDS4 (data providers). While the document is applicable to all such submissions, most of the examples and discussions are presented in a mission/instrument context.

1.3 Document Scope

The DPH introduces the concepts and building blocks around which PDS4 has been designed. It provides instruction on how those building blocks can be constructed in real-life situations and how the simple structures can be assembled into larger aggregations.

PDS4 is built around a very small number of basic data structures; those will be illustrated and a few examples of how they can be extended will be given. But the goal of the DPH is to help the archivist develop basic PDS4 skills rather than to explore the system's flexibility and many possible options. More advanced topics are discussed in [4].

Careful absorption of the DPH coupled with practice should put the reader in a position where s/he can create products from a planetary science instrument and assemble them into a PDS4-compliant archive.

[1] See Appendix A for acronyms and abbreviations. See Section 1.5 for a list of applicable documents.

1.4 Document Overview

PDS4 represents a departure from previous versions of the Planetary Data System. Although it is still an archive of planetary data, it has been designed using contemporary information technology concepts and tools. The system is built around a ‘data model’ that rigorously defines each of its components and the relationships among them. There are only four fundamental data structures, but many extensions are possible — each also rigorously defined. By carefully controlling product definitions and relationships, PDS can accurately track the progress of each product entering the system, compute detailed inventories of holdings, design sophisticated services that users can request to act on subsets of the archive (such as transformations and displays in addition to the expected search and retrieval functions), and connect data products to relevant internal and external information (documentation).

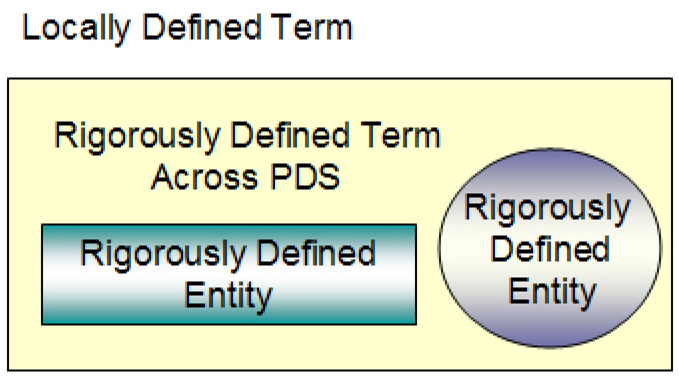
1.4.1 Document Outline

Section 3 of the DPH introduces PDS4 concepts and building blocks and the structures into which they can be assembled. Included is an overview of the PDS4 ‘label’; labels are written in XML, which is also introduced. Section 4 explains how to assemble a bundle …

1.4.2 Document Notation

Notation and terminology in the DPH are consistent with that used in other PDS4 documents [2-4]. Toward that end, a common glossary has been developed (Appendix B).

Rectangular boxes in figures enclose entities and terms that are consistent across PDS. Labeling outside such boxes is not rigorously controlled; readers should be alert to the possibility that such labeling will be different in other documents or in other sections of this document.


Locally Defined Term
Rigorously Defined Term Across PDS
Rigorously Defined Entity
Rigorously Defined Entity

To simplify labeling some figures may use abbreviations. For example <logical identifier> may appear where <logical identifier>content</logical identifier> is the fully expanded XML element (Section 3.1.4.2).

1.5 Applicable Documents

1.5.1 Controlling Document

[1] Planetary Data System (PDS) PDS4 Information Model Specification, Version 0.1.1.1.c

1.5.2 Reference Documents

[2] Planetary Data System Standards Reference, October 2010, Version 4.0.0, JPL D-7669, Part 2

[3] Planetary Science Data Dictionary Document, October 2010, JPL D-7116 Rev. X.

[4] J.S. Hughes, et al., Advanced Topics in PDS4, TBW.

[5] PDS4 XML Tutorial, TBW.

1.6 The PDS 2010 Project

PDS 2010 is a multi-year project to develop and deploy a major modernization of the entire PDS archive and distribution system. The result is referred to as PDS version 4, or simply, PDS4.

This document provides information to assist data providers in the preparation of data for archiving under the PDS4 standards. In previous versions, the Standards Reference included substantial ancillary and tutorial information. Under PDS4 the Standards Reference remains the definitive source for PDS archiving, but the document is designed strictly as a reference. Tutorial information is provided in this and other documents.

Goals of PDS4 include:

- improved efficiency and reduced costs in the data submission process,
- increased robustness and integrity of data in the archive,
- simplied location and retrieval of data from the archive,
- enhanced value added services to end users.

Key principles underlying the development of PDS4 are:

- Data visualization and analysis software change frequently. Formats optimized for such software generally are not optimal for archiving.

  Conversely, data structures optimized for archiving should be simple, rigidly controlled, and projected to be stable for extended periods. Such structures are, in general, less convenient for data visualization and analysis.

- Documents and software in the archive should be handled in the same ways as data are handled.

- The system should identify, locate, and retrieve individual products and identify all associated meta-information and products.

- There should be a few highly constrained, simple data structures which will be stable for decades, allowing development of sophisticated and powerful services and analysis tools.

- Services will include transformations among internally recognized formats and transformations from internal formats to popular formats used by contemporary users.

2.0 PDS4 CONCEPTS AND BUILDING BLOCKS

This section introduces key terms and concepts as they are used within PDS4. Although the choice of terms is intended to be intuitive, readers should understand that the definitions are to be used both rigorously and narrowly within the PDS4 context. For reference, the most important definitions are collected in Appendix B.

2.1 PDS4 Building Blocks – The Primary Pieces

PDS4 is an archive of digital planetary science data — strings of bits, organized in such a way that measurements captured by one observer can be retrieved and used effectively by others.

The strings-of-bits have both structure and meaning. To be interpreted correctly, each string-of-bits must be accompanied by 'metadata' — data about the data — explaining both the structure and meaning. We address structure first.

2.1.1 PDS4 Base Storage Structures

PDS uses four base storage structures. Note that 'scalar' in this context means a single value; a scalar is not necessarily numerical.

- Array_Base - Homogeneous N-dimensional array of scalars
- Table_Base - Repeating heterogeneous record of scalars
- Parsable Byte Stream
- Encoded Stream Base

These structures are rigidly defined by PDS with the key point being that all data archived with the PDS must be stored using these structures.

2.1.1.1 Homogeneous N-dimensional Array of Scalars (Array Base)

A 1024x762 pixel image is a 2-dimensional (2-D) example of an Array Base structure. Each pixel occupies the same number of bits (typically 8 or 16), and the values are interpreted identically. The pixels are organized into fixed rows and columns. A time series of such images taken at one second intervals would be a 3-D example of an Array Base.

2.1.1.2 Repeating Heterogeneous Record of Scalars (Table Base)

A listing of local weather conditions would be an example of a Table Base structure if each record listed measurement location, date and time of the observation, and the measurements themselves. The column giving location (a character string) has a different format and meaning than the columns giving date/time or the numerical values. But every row in the table has the same format and meaning.

2.1.1.3 Parsable Byte Stream

A digital text file or a stock market ticker are examples of the Parsable Byte Stream structure. The American Standard Code for Information Interchange (ASCII) provides a unique translation between letters, digits, punctuation, and a few functions (*e.g.*, TAB and NEWLINE) and the numerical values represented by 8-bit bytes. UTF-8 uses 32-bit words to represent all characters in the world's written languages (with room to spare) — for example, the Cyrillic characters below.

а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я

2.1.1.4 Encoded Byte Stream

Interpreting an Encoded Byte Stream often requires computation in accordance with a recognized international standard. For example, ZIP is a popular compression scheme, and JPEG_2000 can be used for both compression and tiling of image data. The underlying structure and meaning in an encoded stream will not be apparent until after the data have been 'decoded'.

2.1.2 Attributes, Class, and Association

Color is an 'attribute' of clothing, food, and landscapes; it is a property or characteristic that allows us to identify things in our environment and to distinguish some from others. Length, age, and transparency could be other attributes. The physical items in our environment have so many attributes that we don't often think of them in these discrete terms.

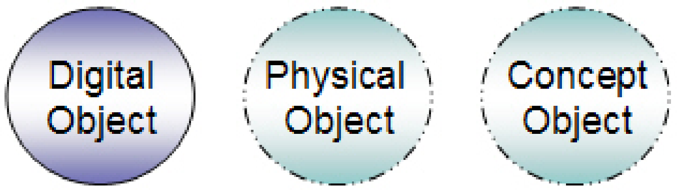
When we organize a digital archive, attributes take on special importance. The size of a file, the number of rows in a table, or the character set chosen (ASCII or UTF-8) are all critical pieces of information in specifying the structure or interpreting the meaning of the archive's contents. 'Color' is not a useful attribute in describing the structure of a digital file; however, ‘wavelength’ or the specification of an imaging system filter may be very helpful in understanding content.

A ‘class’ is the set of attributes which identifies a ‘family.’ A class is generic — a template from which individual members of each family may be constructed. For example, we might decide that all files in our digital archive need to have the following attributes: size, record type, and creation time. Those three attributes then specify a class. Adding a fourth attribute (for example, record length) specifies a *different* class which, in this case, would be a ‘sub-class’ of the first.

An ‘association’ is a defined relationship between classes. It has one direction. For example, a table has one or more columns; “has” is the association.

2.1.3 Data Object

A ‘data object’ is a specific instance of a class. Whereas class is a template, the data object is real. In the archiving context there are three important types of data objects: digital, physical, and conceptual.



Digital objects are everywhere in the archive — for example, the strings-of-bits that comprise tables, images, and documents. Example physical objects include spacecraft and Moon rocks; PDS does not hold physical objects (note dashed outline in figure), but it does include descriptions of them. Example conceptual objects include space missions and PDS discipline nodes; these do not exist as tangible entities (note dashed outline), but they can be described, and the descriptions can be included in the archive.

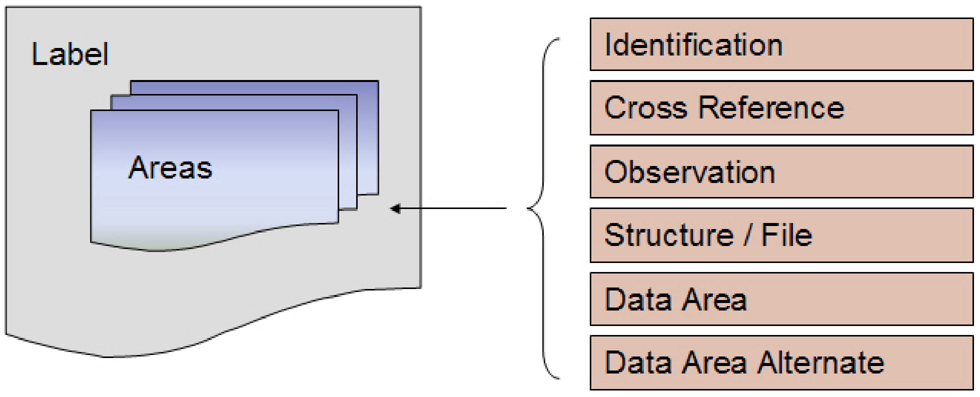
2.1.4 Object Description

For each digital, physical, or conceptual object in, or associated with, the archive we need an ‘object description’. The object description (a data object in its own right) is the collection of metadata (the list of attributes and their values) which describes the structure and meaning of the object to which it is paired. For an image the object description could include the number of lines and samples, the filter used, the time of observation, etc. In PDS4 the object description is expressed in XML as part of a PDS ‘label’.

2.1.4.1 Labels

In the sections above we have explicitly mentioned object descriptions, noted that metadata are used to describe data, and alluded to ‘labels’. A label is the aggregation of all metadata (including, but not limited to, object descriptions and ‘identifiers’) which accompanies one or more strings-of-bits. A label is constructed for a product (Section 3.1.6) or higher level entity; but we introduce it here because of its importance as the home of the object description.

There are six major groupings of information (areas) in a typical label.



Note that in all cases, a label will always be in a file separate from the data file(s) (i.e., the string of bits) being described by the label.

2.1.4.1.1 Identification Area

The identification area is used to uniquely identify each product across the PDS and other archives participating in the federated registry sysem.

The identification area provides unique information for locating one or more products, collections, bundles, etc. identified in the label. The generic identification area includes the following attributes:

- logical identifier
- version identifier
- product class
- title
- alternate identifier (optional)
- one or more alternate titles (optional)
- last modification date/time (optional)

- product subclass (optional)

and an optional association (subject area).

2.1.4.1.2 Cross Reference Area

The cross reference area links the labeled product, collection, bundle, etc. to other products within the archive or its federated partners. The generic cross reference area has no attributes and three optional associations:

- bibliographic reference
- observing system
- reference entry

The cross reference area is the principle mechanism by which products of interest across multiple disciplines are cross referenced / associated. For example, linking calibration, browse, thumbnail, or document products to the primary image product.

2.1.4.1.3 Observation Area

The observation area provides information on the circumstances under which the product, collection, bundle, etc. was acquired. The generic observation area can have up to five attributes:

- start date and time
- stop date and time
- spacecraft clock start count (optional)
- spacecraft clock stop count (optional)
- comment (optional)

and up to two associations:

- mission area (optional)
- node area (optional)

2.1.4.1.4 File Area

The file area provides information on one or more associated data files (i.e., the string of bits being described by the label). Each file has up to 9 attributes:

- file name
- local identifier
- creation date and time (optional)
- file size (optional)
- maximum record bytes (optional)

- number of records (optional)
- MD5 checksum (optional)
- comment (optional)

and a single, required association

- digital object

2.1.4.1.5 Data Area

The data area provides information on at least one associated tagged digital or non-digital object. Possible digital objects include:

- image grayscale
- spectrum 3D
- table character
- table binary
- stream delimited
- SPICE kernel text
- SPICE kernel binary

2.1.4.1.6 Data Area Alternate

The data area alternate provides information on one or more associated tagged digital objects. In many labels this area is not used.

The data area alternate describes/defines the secondary digital object(s) where there is more than one digital object being described in the label. The data area alternate is repeated for each secondary digital object being described/defined.

An example is where a HEADER and a TABLE are co-located in a single data file. The TABLE is considered to be the primary tagged digital object being described and the HEADER is the secondary digial object.

2.1.4.1.7 Label Construction

Labels are constructed using XML and are always derived from a PDS schema. Only PDS approved schemas may be used in deriving a PDS label. PDS labels are XML documents and as such must be XML-compliant in both markup and content (e.g., structure and grammar).

PDS has a tutorial [5] on the general use of XML within the PDS.

2.1.4.2 XML

PDS uses the Extensible Markup Language (XML) for the PDS4 data system. XML is a set of 'open source' rules for encoding documents and data structures in machine-readable form, with special applicability to providing web services. With practice, users can become proficient at reading XML. It is beyond the scope of the DPH to provide an XML tutorial; but we introduce some key concepts in the following paragraphs.

The fundamental structure in XML is the 'tag', which is delimited by "<" and ">".

An 'XML element'[2] begins with <tag> , contains 'content', and ends with </tag>. Particularly simple, 'XML empty element', tags can be represented as <tag/>. For example

```
<date>2009</date>
  <line-feed/>
```

are both XML elements.

XML elements can be nested to create more complicated XML elements. In the following example, identification_area comprises logical_identifier, version_id, title, and last_modification_date (Section 3.1.4.1.1):

```
<identification_area>
  <logical_identifier>URN:NASA:PDS:MPFL-M-IMP-2-EDR-V1.0</logical_identifier>
  <version_id>1.0</version_id>
  <title>MARS PATHFINDER LANDER EXPERIMENT</title>
  <last_modification_date>1998-07-14T00:36:08.000</last_modification_date>
</identification_area>
```

An XML 'schema' (plural schemata) defines the structure of an XML document; it specifies XML elements which must be included (or are optional), their order, and 'parent-child' relationships. XML schemata include attributes and classes from the global dictionary, and the mission and node dictionaries, as needed.

A 'generic' schema has been established for each type of anticipated PDS4 product; these can be tailored by data providers to become 'specific' schemas for their archives. See

http://pds.jpl.nasa.gov/schema/pds4/generic/common

for the current set of generic schemata available for general use.

Labels and other XML documents must be successfully validated against their respective schemata before archives will be accepted.

[2] We distinguish 'XML elements' here from other PDS 'elements' used later, which have different meanings.

2.1.4.2.1 XML Editors

Use of an XML editor simplifies construction and validation of schemata and labels. Two which have been popular during development of PDS4 are oXygen (http://www.oxygenxml.com) and Eclipse (http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.5.2-201002111343/eclipse-SDK-3.5.2-win32-x86_64.zip).

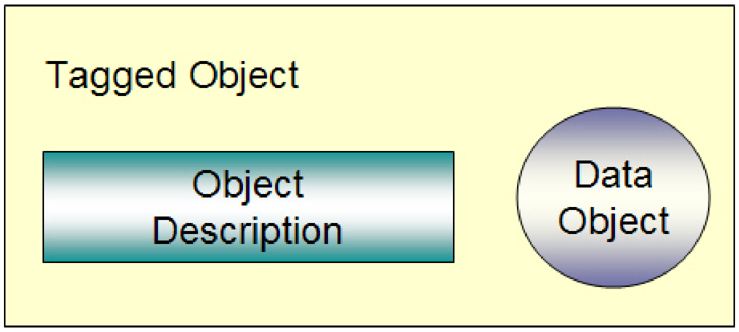
2.1.4.2.2 Label Generators

An XML editor can tailor a generic schema for an application, such as production of labels for a set of spacecraft images. The specific schema is then used as a template in image production software to produce labels for each image.

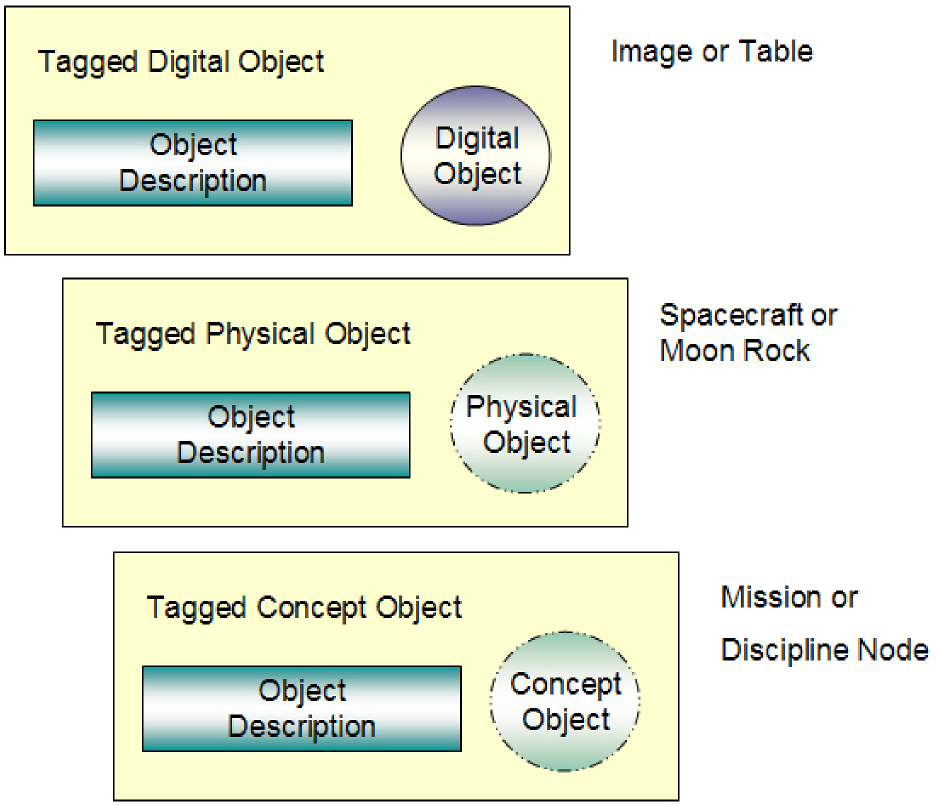
The Python programming language (http://www.python.org/) has gained favor in being able to translate schemata quickly into real labels.

2.1.5 Tagged Object

An object and its associated object description form a 'tagged object'.



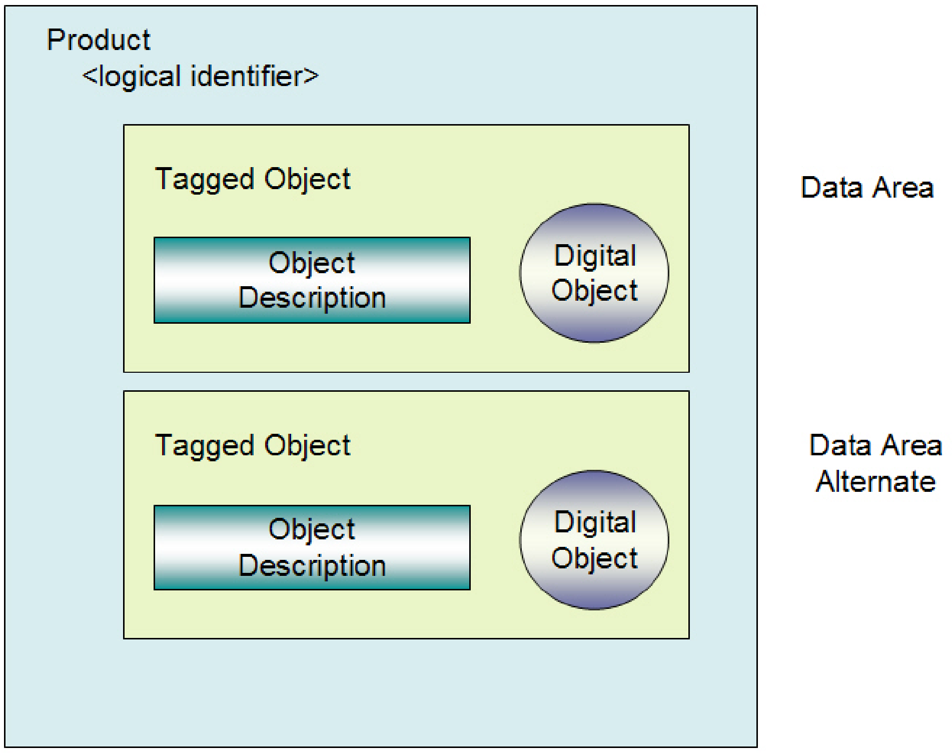
There are three types of Tagged Objects. The type of Tagged Object depends on the type of Data Object being described - Digital Object, or a Physical Object, or a Conceptual Object. The latter two are often called 'tagged nondigital objects.'



In the case of Tagged NonDigital Objects, the associated "data objects" are described in a manner such that they can be referenced and associated with other products in the archive. That is, Tagged NonDigital Objects are 'intrinsic' in the sense that they exist (somewhere) and must be described so that they can be referenced and associated with other products in the archive. However, Physical and Conceptual Objects are not (and cannot be) actually stored in the archive --- because the "data objects" exist as entities that cannot be digitized into the archive.

2.1.6 Product

A product consists of Identification Information and one or more associated tagged objects; the figure below shows a product having two tagged digital objects. Products are 'identifiable' meaning that they can be retrieved from the archive with a single query. In fact, PDS4 uses logical identifiers and version identifiers (Section 2.1.6.1), which allow single query retrievals not only from PDS but also from any other archive participating in its federated registry sysem.



In PDS4 several sub-classes of product are recognized, including:

- Product_Image_Grayscale
- Product_Table_Character
- Product_Table_Binary
- Product _Stream_Delimited
- Product_Document

2.1.6.1 Identifiers

PDS uses identifiers to locate entities (e.g., products, collections, and bundles) internally or externally. Each identified entity is termed an "identifiable". Identifiers are grouped into XML elements called identification areas (see example in Section 2.1.4.2).

2.1.6.1.1 Logical Identifiers

The logical identifier (LID) is unique across the PDS and other federated archives. A LID uniquely identifies the set of all versions of a product. For example, if five versions of an image have been delivered to PDS, the logical identifier allows a user to find all five (or, if different options were exercised in the query, a single preferred version — such as the most recent).

2.1.6.1.2 Version Identifiers

The version identifier (VID) specifies the version of a product. In combination with the logical identifier, the version identifier locates exactly one product in PDS and other federated archives.

Version identifiers have the form:

```
m[.n]
```

where `m` and `n` are both non-negative integers (and `n` is optional), `n` increments by 1 each time there is a 'minor' product revision, `m` increments by 1 each time there is a 'major' revision (and `n` is reset to 0), and the total length is at least 1 but no more than 100 characters.

2.1.6.1.3 LIDVID Identifiers

The LIDVID identifier uniquely identifies a versioned product. The LIDVID is unique across the PDS and other federated archives. The LIDVID is a concatenation of the logical_identifier (LID), two colons, and the version_identifier (VID):

lidvid = lid "::" vid

In the identification_area example above (Section 2.1.4.2), the LIDVID would be

```
URN:NASA:PDS:MPFL-M-IMP-2-EDR::1.0
```

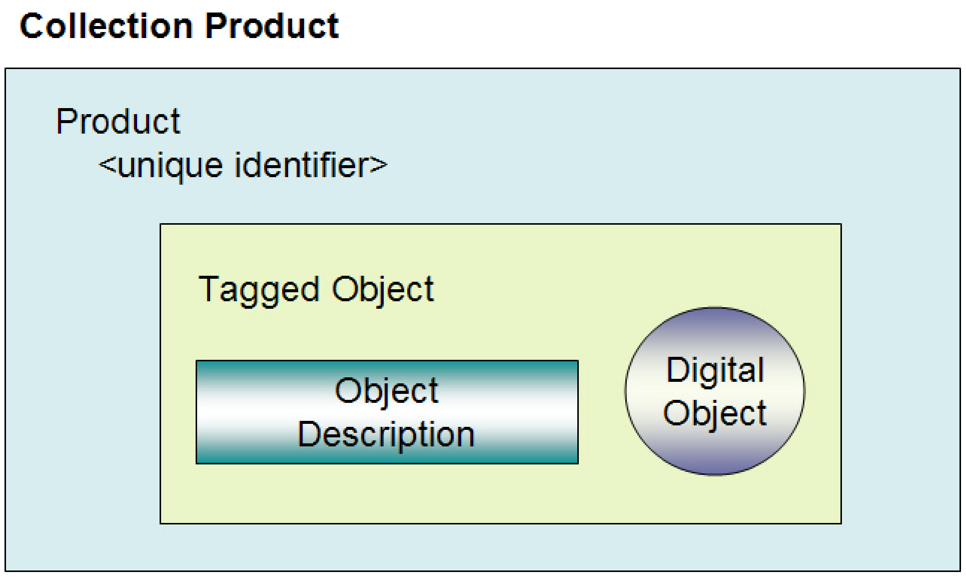
2.1.6.1.4 Local Identifiers

Sometimes one or more identifiers are needed within a label so that product components (objects) can be easily located. In these cases, 'local identifiers' may be used. These are constructed from the class name; in cases where several instances of the same class exist, unique naming conventions should be used to ensure each product component is uniquely identified; and to distinguish among them. Local identifiers are not valid outside the label for which they are defined.

2.1.7 Collection

The next higher level in the organizational hierarchy is the Collection — an inventory of member products, an accompanying label (including an identifier), and (depending on the Collection) the products themselves. The inventory and label are known as a Collection Product. Products of similar type and content are grouped into a collection (i.e., document products are inventoried in a Collection_Document product; context products are inventoried in a Collection_Context product).

In a directory structure the Collection Product is in a directory which has the name of the Collection LID root.



In PDS4 several types of collections may be found in an archive, including:

- Collection_Browse
- Collection_Calibration
- Collection_Context
- Collection_Data
- Collection_Document
- Collection_Geometry
- Collection_SPICE
- Collection_XML_Schema
- Collection_Miscellaneous

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

2.1.7.1 Primary Collection

When delivered to PDS, every product must be associated with a single primary collection. The data provider defines the association.

The Primary Collection Product identifies member products by LIDVID.

In a directory structure, the Primary Collection member products are in a single subdirectory, or subdirectories thereof, which is parallel to the Collection Product.

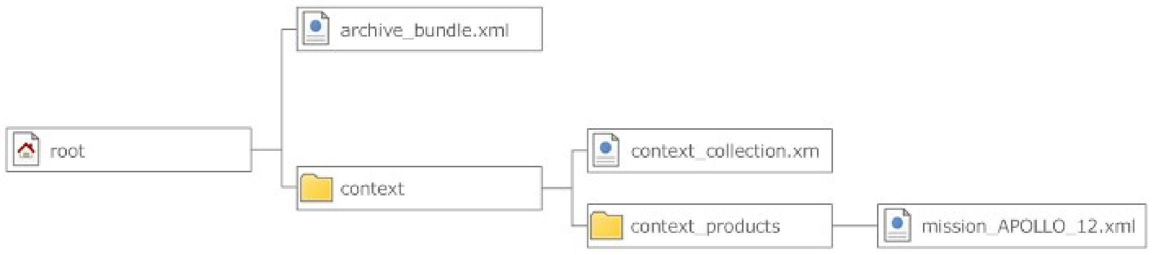
2.1.7.2 Secondary Collection

Products which are already in the archive (and, therefore, associated with a primary collection) may later be associated with other, secondary collections. For example, a set of Mars Reconnaissance Orbiter images collected during observations in 2009 may be assigned to the 'MRO_2009' primary collection. Another investigator may wish to study all Mars images covering 0-5 S latitude 200-205 E longitude; some of the MRO_2009 images would qualify, but so would MRO images from other years, as would images from other spacecraft imaging systems.

A Secondary Collection Product identifies member products by LID, VID, or LIDVID. Ordinarily a Secondary Collection would not include products; for delivery to a user, however, PDS may create a temporary subdirectory parallel to the Secondary Collection Product in which products are stored before transfer.

2.1.7.3 Collection Inventory

Each collection product is comprised of a collection label and a collection inventory. The collection inventory, a manifest of the products that comprise the collection product, is a two column table where each row of the table describes one of the products in the collection. The first column of the table is the LIDVID of the product. The second column of the table is the file_specification_name of the product (which is the name and location of the product relative to the location of the collection product).



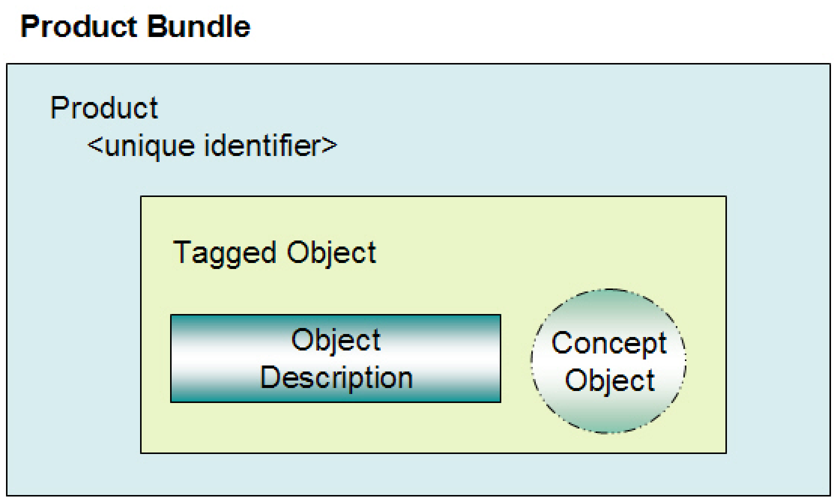
In the above diagram for the mission_APOLLO_12 context product:

1. The LIDVID would be the value of the <logical_identifer> in the mission_APOLLO_12.xml file.
2. The file_specification_name would be “context_products\mission_APPOLO_12.xml”.

Note that the backslash (“\”) is used to denote directories and to delineate subdirectories.

2.1.8 Bundle

Just as a Collection groups products, a Bundle groups Collections. The Bundle comprises a Product Bundle (a labeled inventory) and subdirectories for each Collection. Because the number of Collections is small, the inventory and label may be combined efficiently into a single XML file, effectively making the inventory a conceptual object.



In PDS4 there are no Bundle sub-classes.

An example can be found at the following:

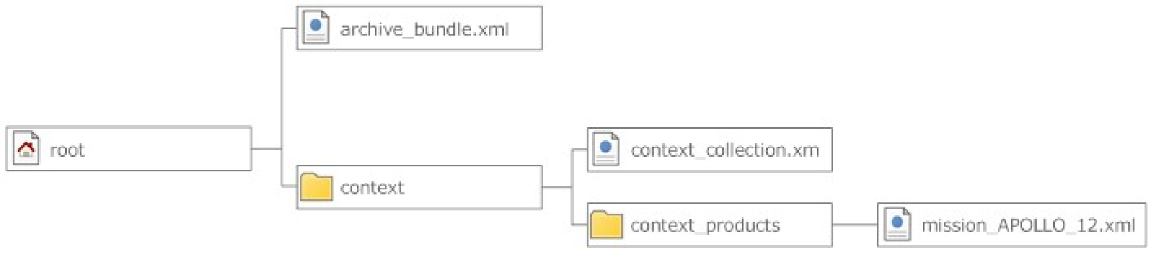
http://pds.jpl.nasa.gov/repository/pds4/examples/

2.1.8.1 Structure of Bundles

A Product Bundle identifies all of its member collections (collection_browse, collection_calibration, collection_data, etc). The included collections are each described by a Bundle_Member_Entry XML element, as in the example below:

```
<Bundle_Member_Entry>
    <file_specification_name >Collection_SPICE</file_specification_name>
    <lid_reference>URN:NASA:PDS:Collection_SPICE</lid_reference>
    <reference_association_type>has_association</reference_association_type>
</Bundle_Member_Entry>
```

The following steps describe how the values in the Bundle_Member_Entry XML element are derived.



In the above diagram for the mission_APOLLO_12 context product:

1. <file_specification_name> is the relative path from Bundle.xml to the directory holding the Collection_Product label. The file_specification_name would be "context\mission_APOLLO_12.xml".

2. <lid_reference> is the value of the <logical_identifier> in the Collection_Product label.

3. <reference_association_type> would be "has_context_collection".

   Note that the value for <reference_assocation_type> is dependent upon the type of collection being referenced. For example, a browse collection would have a <reference_association_type> of "has_browse_collection".

Type of Collection	<reference_association_type>
browse	has_browse_collection

calibration	has_calibration_collection
data	has_data_collection
document	has_document_collection
geometry	has_geometry_collection
spice	has_spice_collection
xml_schema	has_xml_schema_collection
context	has_context_collection
“other”	has_miscellaneous_collection

2.2 The PDS Registry

Products, collections, and bundles are related to one another through the registry system, using metadata supplied in labels. The registry can be viewed as a set of linked data bases, at least one of which keeps track of PDS data holdings.

2.2.1 Registry Use of Identification XML Elements

The label Identification_Area (Section 2.1.4.1.1) contains naming and origination information which the registry captures and stores, keyed to the logical_identifier (Section 2.1.6.1.1) and/or the LIDVID (Section 2.1.6.1.3). An example Identification_Area_Product, is shown below.

```
<Identification_Area_Product>
   <logical_identifier>
      URN:NASA:PDS:MGS-M-RSS-5-TPS-V1.0:TPS:4147E13A.TPS
   </logical_identifier>
   <version>1.0</version_id>
   <product_class>Product_Table_Character</product_class>
   <title>4147E13A.TPS</title>
   <last_modification_date_time>
      2005-05-06T04:27:43
   </last_modification_date_time>
   <Subject_Area>
      <instrument_host_name>MARS GLOBAL SURVEYOR</instrument_host_name>
      <instrument_name>RADIO SCIENCE SUBSYSTEM</instrument_name>
      <target_name>MARS</target_name>
   </Subject_Area>
</Identification_Area_Product>
```

2.2.2 Registry Use of Cross Reference XML Elements

The Cross_Reference_Area_Product contains links to other products, collections, or bundles in PDS or any other archive served by its federated registry. The primary function of the Cross_Reference_Area_Product of a PDS4 label is to be the vehicle by which various types of products can be “linked / associated” with / to the “identifiable” which in this case is the primary product being described by the product label.

The following diagram illustrates how the “Cross_Reference_Area_Product” is used to make associations to the primary product via the unique identifier of the primary product.

The unique identifier for the primary product, the product being described by the product label, is the value of the logical_identifier as specified in the Identification_Area_Product.

```
<Product_Identification_Area>
    <logical_identifier>URN:NASA:PDS:instrument_host.24COL</logical_identifier>
    <version_id>v1.0</version_id>
    <object_type>Product_Instrument_Host</object_type>
    <title>24-COLOR SURVEY</title>
    <last_modification_date_time>
      2004-04-02T00:00:00.00
    </last_modification_date_time>
    <primary_collection_reference>
      URN:NASA:PDS:context_collection.instrument_host::v1.0
    </primary_collection_reference>
    <Subject_Area>
        <instrument_host_name>24-COLOR SURVEY</instrument_host_name>
    </Subject_Area>
</Product_Identification_Area>
```

The set of products that are to be linked to / referenced by the primary product are identified by the lidvid_reference as specified in the Product_Reference_Entry class which is a subclass of the Cross_Reference_Area_Product class.

```
<Cross_Reference_Area_Product>
    <Product_Reference_Entry>
        <lid_reference>URN:NASA:PDS:investigation.PHOENIX</lid_reference>
        <reference_association_type>has_investigation</reference_association_type>
    </Product_Reference_Entry>
    <Product_Reference_Entry>
        <lid_reference>URN:NASA:PDS:instrument_host.PHX</lid_reference>
        <reference_association_type>has_instrument_host</reference_association_type>
    </Product_Reference_Entry>
    <Product_Reference_Entry>
        <lid_reference>URN:NASA:PDS:instrument.TT+PHX</lid_reference>
        <reference_association_type>has_instrument</reference_association_type>
    </Product_Reference_Entry>
    <Product_Reference_Entry>
        <lid_reference>URN:NASA:PDS:target.MARS</lid_reference>
        <reference_association_type>has_target</reference_association_type>
    </Product_Reference_Entry>
    <Product_Reference_Entry>
        <lid_reference>URN:NASA:PDS:node.ATMOS</lid_reference>
        <reference_association_type>has_node</reference_association_type>
    </Product_Reference_Entry>
</Cross_Reference_Area_Product>
```

In the above Cross_Reference_Area_Product example, there are five Product_Reference_Entry classes which identify and provide an association to the primary product.   In the above, the primary product has associations to:

1. An investigation: `URN:NASA:PDS:investigation.PHOENIX`
2. An instrument_host: `URN:NASA:PDS:instrument_host.PHX`
3. An instrument: `URN:NASA:PDS:instrument.TT+PHX`

4. A Target: URN:NASA:PDS:target.MARS
5. An node: `URN:NASA:PDS:node.ATMOS`

Each association is uniquely identified by the "lidvid_reference". And, each type of association is identified by the "reference_association_type".

Given the "lidvid_reference", the registry will be able to locate the product and all products that are linked / have associations with the product.

The full set of association types are listed in the following table.

Product_Browse	has_browse
Product_Calibration	has_calibration
Product_Geometry	has_geometry
Product_PDS_Affiliate or Product_PDS_Guest	has_personnel
Product_SPICE	has_spice
Product_Target	has_target
Product_Thumbnail	has_thumbnail
Product_Document	has_document
Product_Investigation	has_investigation
Product_Instrument_Host	has_instrument_host
Product_Instrument	has_instrument
Product_Node	has_node
Product_Resource	has_resource

2.3 Miscellaneous Label Topics

TBD

2.3.1 Elusive Values

2.3.1.1 Elusive Values in Labels

During creation of product labels, appropriate values for some attributes may not exist. In such cases, PDS4 provides symbolic values, each of which is appropriate for specific circumstances.

- “not applicable” indicates that the attribute is not relevant; no value is known and knowledgeable users would not expect to find one.
- “unknown” indicates that the value for the data element is not known and never will be known.
- “temporarily not known” is used when the value is *temporarily* unknown. It indicates that the data provider recognizes that a value should be inserted, but the true value is not readily available.

There are a limited set of attributes where the values “not applicable”, “unknown”, and “temporarily not known” can be specified (*e.g.*, time values, numeric values).

2.3.1.2 Elusive Values in Data Products

A second approach exists for numeric fields. Numeric fields should not contain character strings (*e.g.*, “not applicable”); but an otherwise unexpected numeric value can be used to flag special circumstances. The Special_Constants class, in conjunction with the following XML tags, is used to provide a set of such values:

```
<error_constant>
<invalid_constant>
<missing_constant>
<not_applicable_constant>
<saturated_constant>
<unknown_constant>
```

For example, if a two-digit value is “unknown” — that is, the value is not known and never will be known — a value outside the normal range can be chosen to indicate “unknown” — *e.g.*, 999. To flag that the data were invalid (in some respect), a different value (*e.g.*, -999) could be chosen to indicate that second condition. The XML Special_Constants element would then look like

```
<Special_Constants>
    <invalid_constant>-999</invalid_constant>
    <unknown_constant>999</unknown_constant>
</Special_Constants>
```

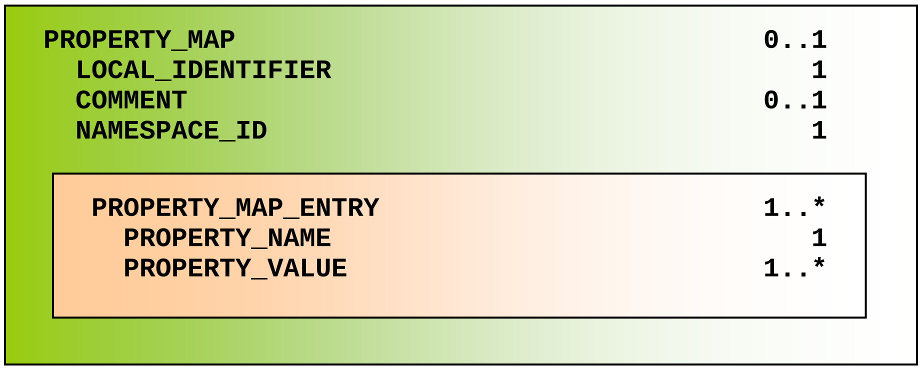
The individual uses for the above Special_Constants elements are defined in the Data Dictionary [3].

2.3.2 Cardinality

Cardinality of a set is the number of elements in the set. For example, the set A= {1,5,10} contains 3 elements, and therefore A has a cardinality of 3. PDS4 uses cardinality to specify the number of attribute values or associations expected in labels; it also provides a shorthand for indicating whether ‘children” are optional or required. See the table below for example values and interpretations:

Cardinality	***Desciption***
0..1	Within the context of the parent, the child may optionally exist as a single non-repeating instance
0..*	Within the context of the parent, the child may optionally exist as an unbounded repeating instance
1	Within the context of the parent, the child must exist once and only once
1..*	Within the context of the parent, the child must exist at least once
2	Within the context of the parent, the child must exist twice and only twice
2..*	Within the context of the parent, the child must exist at least twice

The diagram below illustrates the parent-child relationship using cardinality nomenclature.



```
PROPERTY_MAP                                       0..1
  LOCAL_IDENTIFIER                                    1
  COMMENT                                          0..1
  NAMESPACE_ID                                        1

   PROPERTY_MAP_ENTRY                              1..*
     PROPERTY_NAME                                    1
     PROPERTY_VALUE                                1..*
```

1. The PROPERTY_MAP class comprises a single child class, the PROPERTY_MAP_ENTRY class.
2. The PROPERTY_MAP_ENTRY class must exist once but may exist many times within the context of the parent PROPERTY_MAP class.
3. The parent PROPERTY_MAP class comprises two required non-repeating attributes (LOCAL_IDENTIFIER and NAMESPACE_ID) and a single optional non-repeating attribute (COMMENT).
4. The PROPERTY_MAP_ENTRY class comprises a single required non-repeating data element (PROPERTY_NAME)

3.0 ASSEMBLING A BUNDLE

All data archived with the PDS must be delivered in a Bundle. This section describes the Collections which are required and the process that is used to create such a Bundle.

3.1 Structure of the Bundle

Figure 3.1 shows the structure of a Bundle, representative Collections, and their relationships.

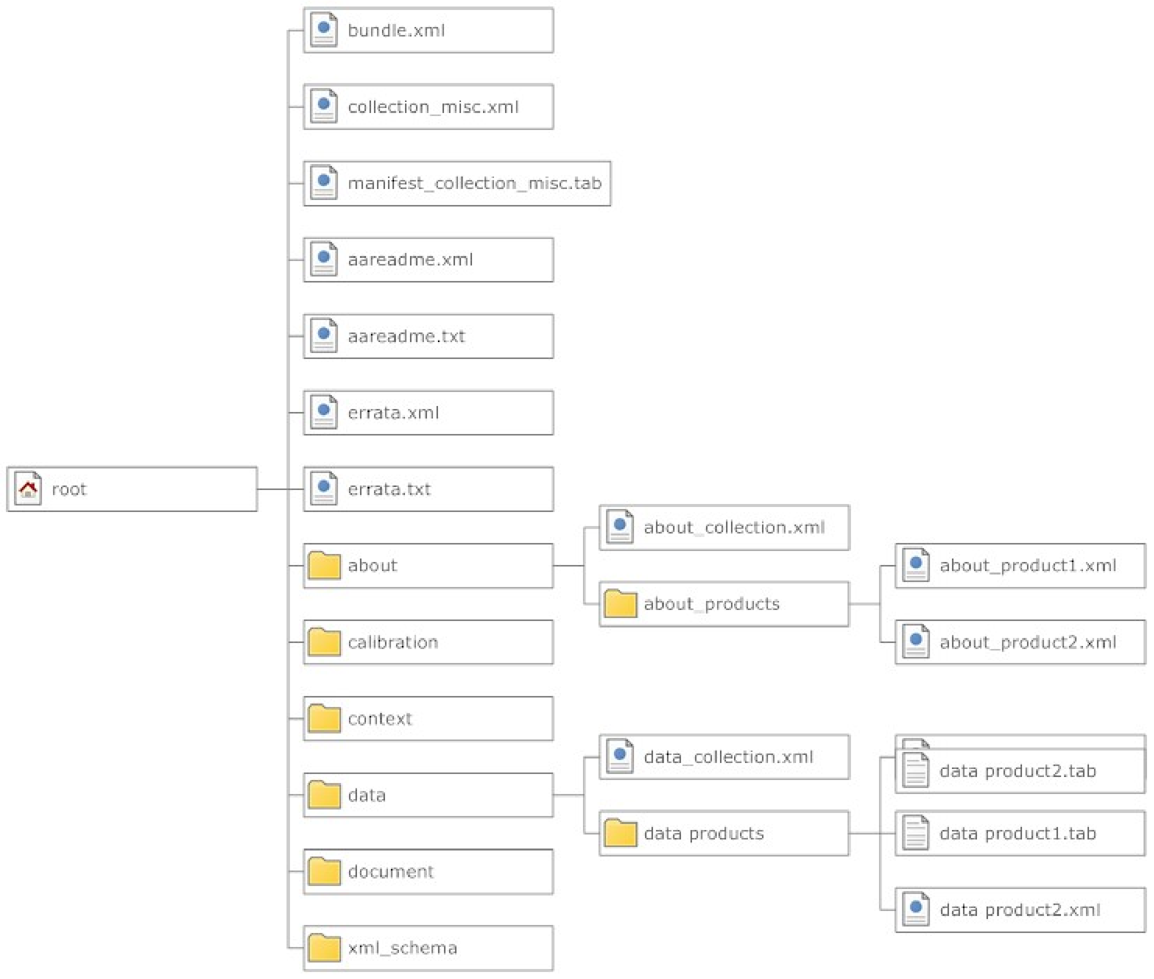


Figure 3-1. Physical Structure of an Archive Bundle (abbreviated)

Figure 3-2 shows the root of a Bundle, including an example of each type of possible collection. Cardinality of each Collection and whether the Collection name can be chosen by the data provider are shown at the bottom of each box. The Bundle_Product, a file with the name bundle.xml, is required (cardinality 1). A Document_Collection is optional (cardinality "0..1"); if present, it must be present in a directory "document".

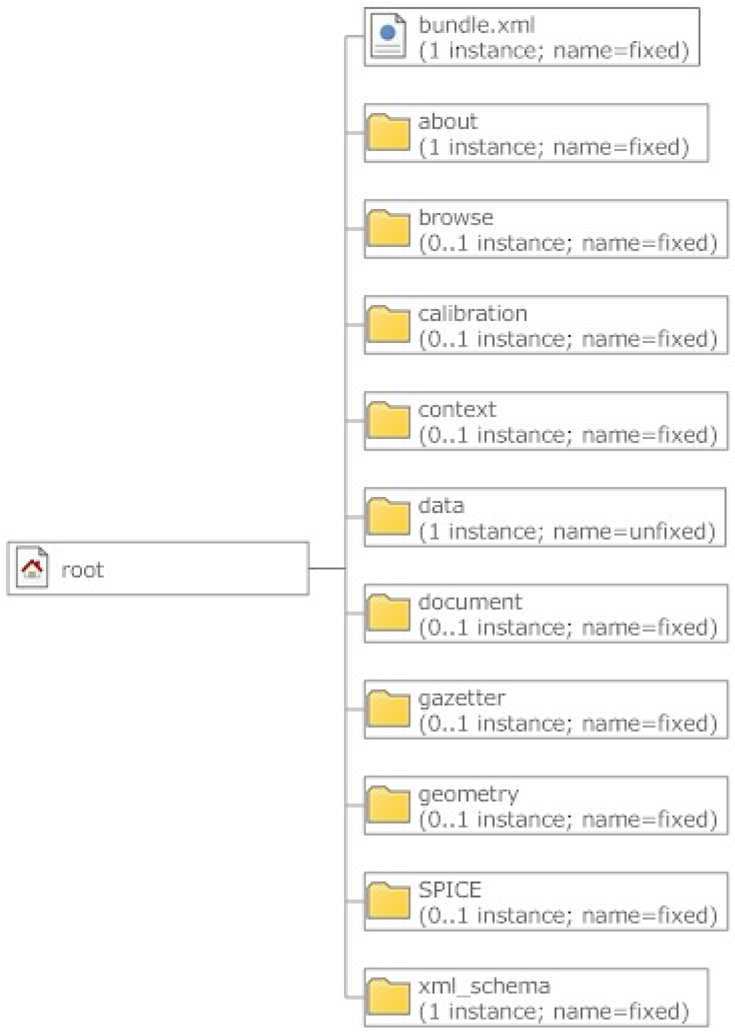

root
bundle.xml
(1 instance; name=fixed)
about
(1 instance; name=fixed)
browse
(0..1 instance; name=fixed)
calibration
(0..1 instance; name=fixed)
context
(0..1 instance; name=fixed)
data
(1 instance; name=unfixed)
document
(0..1 instance; name=fixed)
gazetter
(0..1 instance; name=fixed)
geometry
(0..1 instance; name=fixed)
SPICE
(0..1 instance; name=fixed)
xml_schema
(1 instance; name=fixed)

Figure 3-2. Top-level Root Structure of an Archive Bundle

3.1.1 Bundle Product

Cardinality: 1 (required)
Naming: fixed (bundle.xml)

This XML file uniquely defines the Bundle; it lists every member Collection, first by LIDVID and second by file_specification_name.

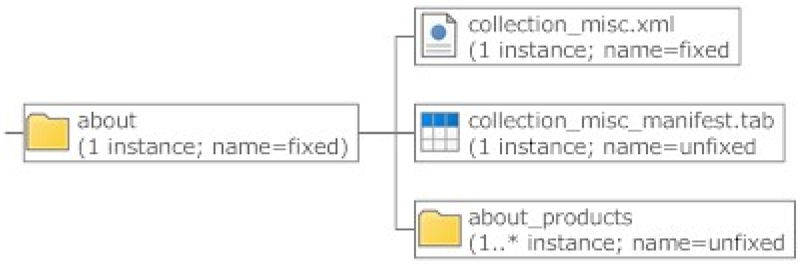
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.2 about directory

Cardinality: 1 instance required
Naming: fixed

This directory contains files that provide information that facilitiates the use of the archive. The following figure describes the structure and contents of the "about" directory.



Note that collection_miscellaneous is used as a collector of about products.

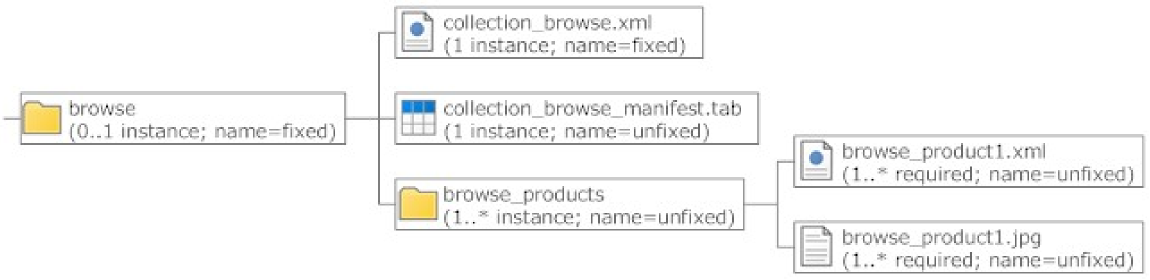
An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.3 Browse Collections and Directories

Cardinality: 0..* (optional and unlimited)
Naming: unfixed

Browse Collections contain ‘quick-look’ products that facilitate use of the archive. Some data providers choose Browse directory structures that parallel the directories in corresponding Data Collections.


browse
(0..1 instance; name=fixed)
collection_browse.xml
(1 instance; name=fixed)
collection_browse_manifest.tab
(1 instance; name=unfixed)
browse_products
(1..* instance; name=unfixed)
browse_product1.xml
(1..* required; name=unfixed)
browse_product1.jpg
(1..* required; name=unfixed)

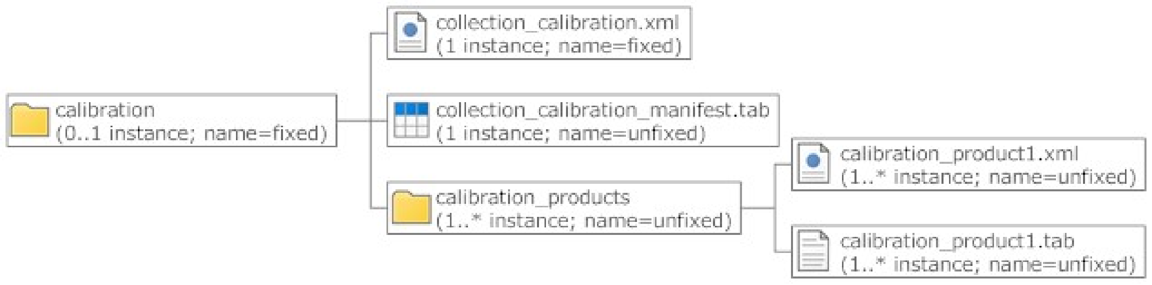
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.4 Calibration Collections and Directories

Cardinality: 0..* (optional and unlimited)
Naming: unfixed

Calibration Collections contain calibration data and other files necessary for the calibration of the data products. Some data providers choose Calibration directory structures that parallel the directores in corresponding Data Collections.


calibration
(0..1 instance; name=fixed)
collection_calibration.xml
(1 instance; name=fixed)
collection_calibration_manifest.tab
(1 instance; name=unfixed)
calibration_products
(1..* instance; name=unfixed)
calibration_product1.xml
(1..* instance; name=unfixed)
calibration_product1.tab
(1..* instance; name=unfixed)

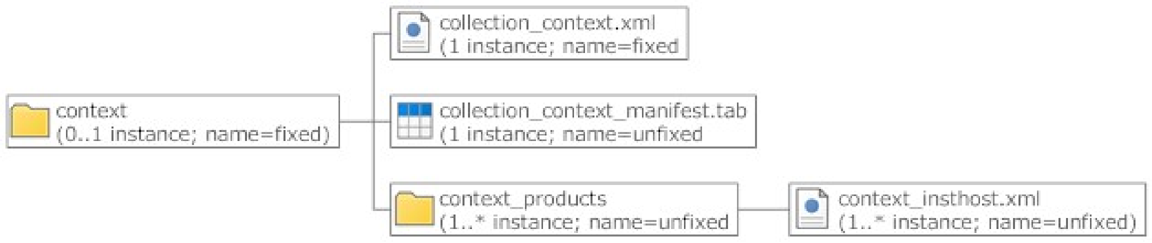
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.5 Context Collection and Directory

Cardinality: 0..1 (optional, but no more than one)
Naming: fixed (CONTEXT)

A Context Collection contains all context products associated with an archive. These are the products identified in the Product Cross Reference Area of the data products in the archive. If present, the Contrext Collection is in the context directory.


context
(0..1 instance; name=fixed)
collection_context.xml
(1 instance; name=fixed
collection_context_manifest.tab
(1 instance; name=unfixed
context_products
(1..* instance; name=unfixed
context_insthost.xml
(1..* instance; name=unfixed)

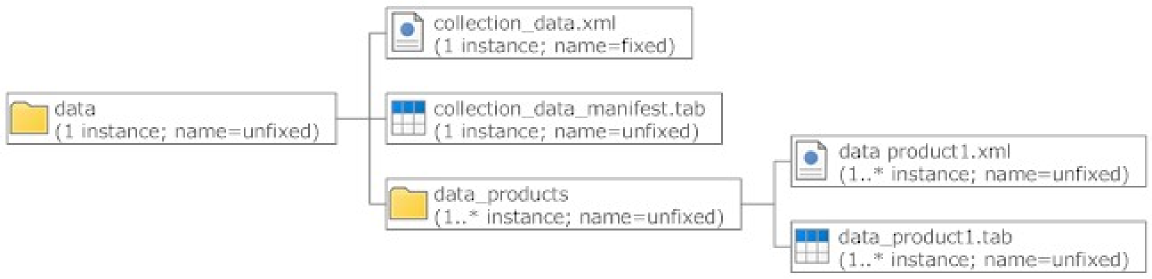
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.6 Data Collections and Directories

Cardinality: 0..* (optional and unlimited)
Naming: unfixed

Data Collections contains the archive data products. Some data providers choose to organize data into subdirectories to prevent over-crowding and facilitate archive organization.


data
(1 instance; name=unfixed)
collection_data.xml
(1 instance; name=fixed)
collection_data_manifest.tab
(1 instance; name=unfixed)
data_products
(1..* instance; name=unfixed)
data product1.xml
(1..* instance; name=unfixed)
data_product1.tab
(1..* instance; name=unfixed)

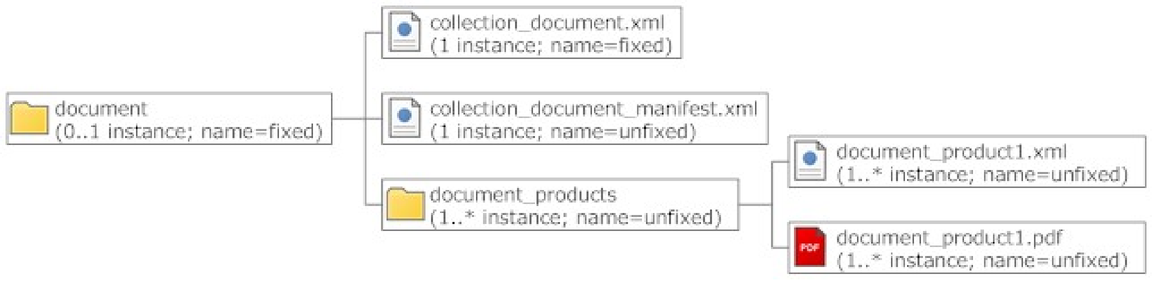
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.7 Document Collections and Directory

Cardinality: 0..1 (optional, but not more than one)
Naming: fixed (DOCUMENT)

A Document Collection contains document products that provide documentation and supplementary and ancillary information to assist in understanding and using the data products in the Bundle. The documentation may describe the mission, spacecraft, instrument, and data. It may include references to science papers published elsewhere. .


document
(0..1 instance; name=fixed)
collection_document.xml
(1 instance; name=fixed)
collection_document_manifest.xml
(1 instance; name=unfixed)
document_products
(1..* instance; name=unfixed)
document_product1.xml
(1..* instance; name=unfixed)
document_product1.pdf
(1..* instance; name=unfixed)

An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

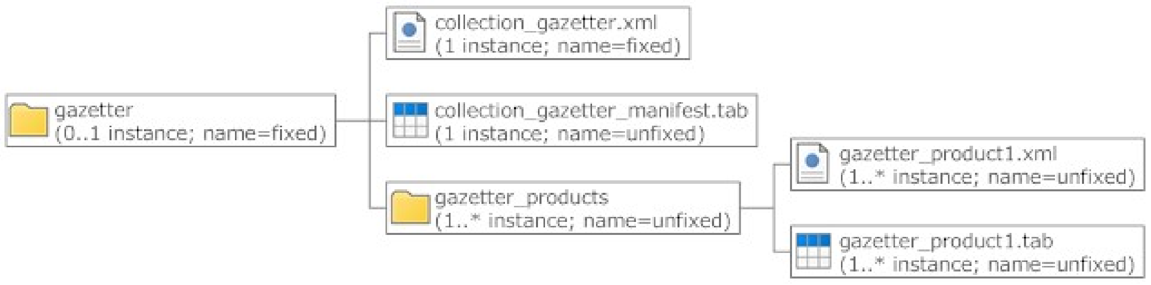
Notes:

1. All files comprising a document product, as described in the product's label, must be in the directory containing the label or in subdirectories of that directory. If not in the same directory, the label must show the relative path from the root of the DOCUMENT directory to each.
2. When multiple files comprise a document product, the parent file is listed first in the label.

3.1.8 Gazetteer Collection and Directory

Cardinality: 0..1 (optional, but not more than one)
Naming: fixed (GAZETTEER)

A Gazetteer Collection contains information about all the named features on a target body associated with the data products in the bundle. "Named features" are those the International Astronomical Union (IAU) has named and approved.


gazetter
(0..1 instance; name=fixed)
collection_gazetter.xml
(1 instance; name=fixed)
collection_gazetter_manifest.tab
(1 instance; name=unfixed)
gazetter_products
(1..* instance; name=unfixed)
gazetter_product1.xml
(1..* instance; name=unfixed)
gazetter_product1.tab
(1..* instance; name=unfixed)

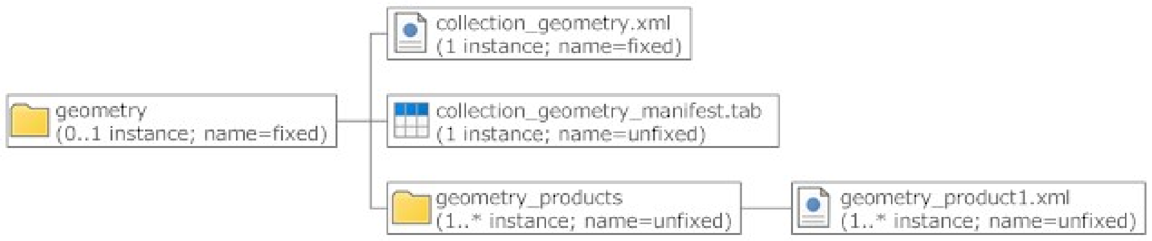
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.9 Geometry Collection and Directory

Cardinality: 0..1 (optional, but not more than one)
Naming: fixed (GEOMETRY)

A Geometry Collection contains non-SPICE information that describes or is related to the observing geometry for the data products in the Bundle. A Geometry Collection is appropriate for observations that were conducted without SPICE or for supplementary information which was derived using SPICE..


geometry
(0..1 instance; name=fixed)
collection_geometry.xml
(1 instance; name=fixed)
collection_geometry_manifest.tab
(1 instance; name=unfixed)
geometry_products
(1..* instance; name=unfixed)
geometry_product1.xml
(1..* instance; name=unfixed)

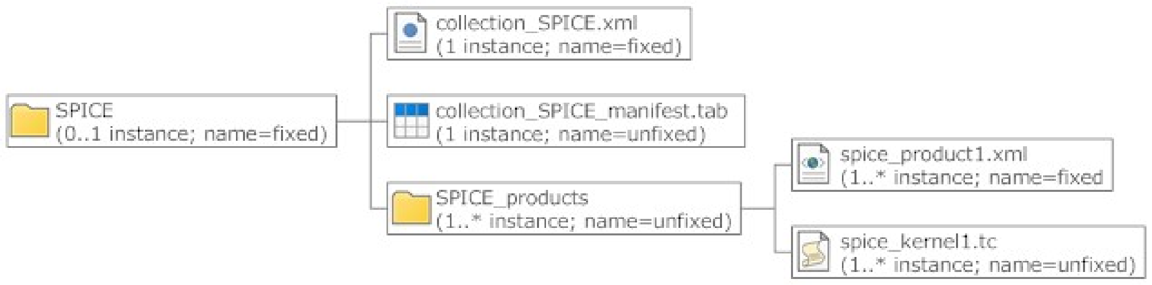
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.10 SPICE Collection and Directory

Cardinality: 0..1 (optional, but not more than one)
Naming: fixed (SPIC)

A SPICE Collection contains SPICE kernels relevant to the data products in the Bundle, SPICE software, and SPICE documentation. The SPICE system includes a suite of software, mostly in the form of subroutines, that may be incorporated into users' application programs to read SPICE data files and compute observation geometry such as altitude, latitude/longitude, and lighting angles.


SPICE
(0..1 instance; name=fixed)
collection_SPICE.xml
(1 instance; name=fixed)
collection_SPICE_manifest.tab
(1 instance; name=unfixed)
SPICE_products
(1..* instance; name=unfixed)
spice_product1.xml
(1..* instance; name=fixed
spice_kernel1.tc
(1..* instance; name=unfixed)

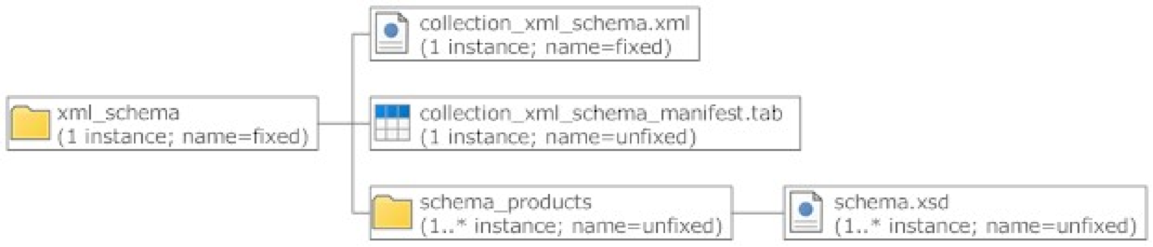
An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.11 XML_Schema Collection and Directory

Cardinality: 1 (one and only one Collection is required)
Naming: fixed (XML_SCHEMA)

The XML Schema Collection contains every schema that has been used in creating the Bundle, its Collections, and their Products.


xml_schema
(1 instance; name=fixed)
collection_xml_schema.xml
(1 instance; name=fixed)
collection_xml_schema_manifest.tab
(1 instance; name=unfixed)
schema_products
(1..* instance; name=unfixed)
schema.xsd
(1..* instance; name=unfixed)

An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.12 Miscellaneous Collections and Directories

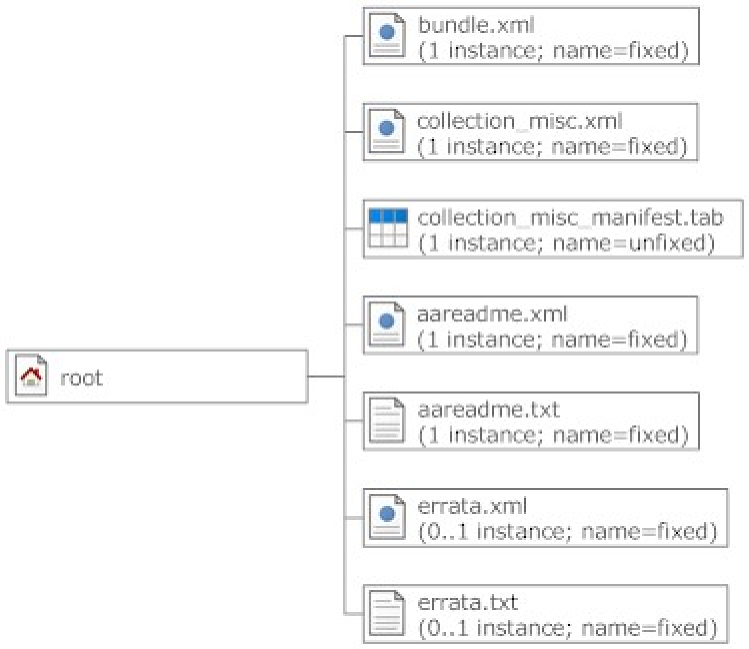
A Miscellaneous Collection contains information which does not conveniently fall into other, named Collections and which is relevant to Products in the Bundle (e.g., about products).

An example can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.13 ROOT Directory Files and Collection

There are a number of required and optional files contained in the ROOT directory.



3.1.13.1 bundle XML Product

See Section 3.1.1 for a description of bundle.xml.

3.1.13.2 aareadme XML Product

Cardinality: 1 instance required
Naming: fixed (aareadme.xml)

This XML document describes the aareadme.txt file. The name, aareadme.xml, has been chosen so that it will be listed (almost) first in an alphabetical directory listing.

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.13.3 aareadme.txt

Cardinality: 1 instance required
Naming: fixed (aareadme.txt)

This file describes the contents and organization of the associated archive, contains general instructions for its use, and has contact information of those most familiar with the archive. The name, aareadme.txt, has been chosen so that it will be listed first in an alphabetical directory listing.

3.1.13.4 errata XML Product

Cardinality: 0..1 instance
Naming: fixed (errata.xml)

This XML document describes the errata.txt file.

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.13.5 errata.txt

Cardinality: 1 instance required
Naming: fixed (errata.txt)

This file identifies and describes errors and/or anomalies found in the archive.

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.13.6 collection_miscellaneous XML Product

Cardinality: 1 instance required
Naming: fixed (collection_misc.xml)

This XML document describes the files contained in the root directory. The collection_misc.xml with the collection_misc_manifest.tab are the collection product.

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

3.1.13.7 collection_miscellaneous Inventory

Cardinality: 1 instance required
Naming: unfixed

The collection inventory, a manifest of the products that comprise the collection miscellaneous product, is a two column table where each row of the table describes one of the products in the collection. The first column of the table is the LIDVID of the product. The second column of the table is the file_specification_name of the product (which is the name and location of the product relative to the location of the collection product). See Section 2.1.7.3 for additional information.

3.2 Assembling the Collections

Each Collection in a Bundle is distinguished by having products of a different 'type'; the distinctinguishing characteristics are established by the data provider and may include processing level (raw vs derived), subject (documents vs data), source (different instruments), format (ASCII vs binary), time (2009 vs 2010), or other criteria. The recognized subclasses of Collection are outlined above; Miscellaneous Collections contain information which does not fit easily into one of the other sub-classes.

Each Collection is defined by a Collection_Product — a table listing every product in the collection with information on how to locate the product and an XML label for the table. The Collection Product is in a directory which has the name of the Collection LID root. Member products are in a subdirectory parallel to the Collection Product, which may itself be divided into subdirectories.

An example can be found at the following url:

http://pds.jpl.nasa.gov/repository/pds4/examples/

4.0 PDS4 DATA REPRESENTATION

Data can be an elusive concept. Data may exist in some storage format on some disk somewhere, on paper somewhere else, in active memory on some server, or transmitted along some wire between two computers. All these can still represent the same data. That is, there is an important distinction to be made between the data and its representation. The data consist of numbers: abstract entities that usually represent measurements of something, somewhere. Data also consist of the relationships between those numbers, as when one number defines a time at which some quantity was measured.

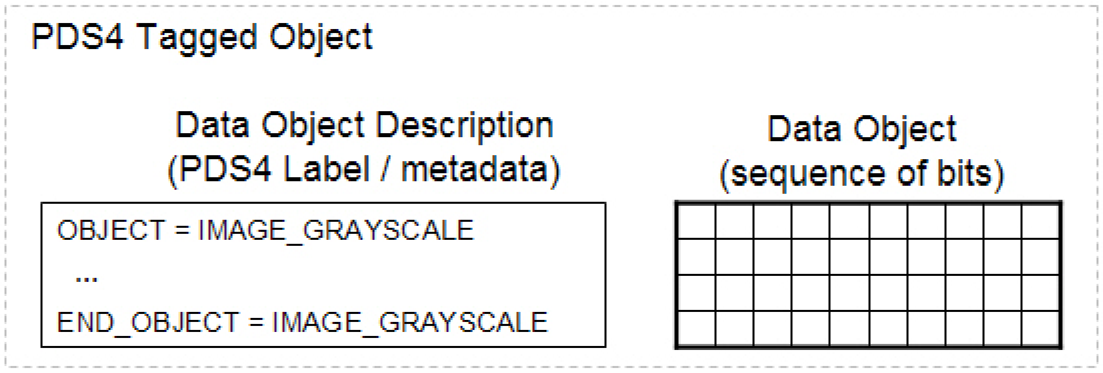
The abstract existence of data is in contrast to its concrete representation, which is how the data is viewed, manipulated, and stored. Data can be stored as BCD numbers in a file on a disk, or as twos-complement integers in the memory of some computer, or as numbers printed on a page. It can be stored in netCDF, HDF, JGOFS, a relational database and any number of other digital storage forms.

The PDS specifies a particular representation of data, to be used in archiving that data. This “archival” representation distinguishes it from the representations used in some computer’s memory (i.e., how the data is stored or represented on either the sending or receiving computer; or the transmission format used to communicate between the two servers).

For this document, we identify two special types of objects -- the "data object" and the "data object description." The data object contains "data," and (by itself) is not otherwise constrained. The data object description contains information about another object, such as a data object. By linking a data object with a data object description, we create a pair which includes both the data and enough information that we can start to read and interpret the bits --- a PDS Tagged Object.

A data object description can (and often does) exist without being physically accompanied by another object. The object it describes may not be physical (e.g., a space mission which, although it has physical components, is itself a concept) or it may not be practical to include the physical object (e.g., the planet Saturn).

Note that within the context of this document, of three types of data objects (digital, conceptual, and physical), we will only address “digital data objects”.



At its simplest, a PDS4 Tagged Object consists of a PDS4 Data Object Description (e.g., a PDS4 Label) and a “digital” Data Object (e.g., sequence of bits) that are described by the metadata resident in the PDS4 Label. The Data Object Description describes both the physical and logical structure of the referenced Data Object.

4.1 PDS4 Data Structures

PDS4 defined four new basic types of data structures for the purposes of describing data objects. All current PDS4 digital object classes fall into one of the four basic data structures.

1. Array_Base - Homogeneous N-dimensional array of scalars

Homogeneous N-dimensional array of scalars -- describes a collection of "items" of the same type. Every "item" takes up the same size block of memory, and all blocks are interpreted in exactly the same way (i.e., the number of “items” in an array is fixed by that specified by the size of its dimension). How each "item" in the array is to be interpreted is specified by a separate data-type class, of which one is associated with every array (i.e., the “items” in an array are represented by an identical storage format – MSB_INTEGER_4_BYTE, MSB_INTEGER_2_BYTE, etc).

An instance of the Array_Base class consists of a collection of contiguous one-dimensional segments of memory (owned by the array), combined with an indexing scheme that maps the "items". How many bytes in each "item" and how the bytes are interpreted is defined by the data-type class associated with the array (i.e., basic constraints on storage order, element types, and maximum number and length of axes are defined by the data-type class).

Example Classes:
- Image_Grayscale
- 3D Image

2. Table_Base - Heterogeneous repeating record of scalars

Heterogeneous repeating record of scalars -- describes a collection of "items" where the "items" characteristics may vary within a row of "items". Every column of "items" takes up the same size block of memory, and all blocks are interpreted in exactly the same way (i.e., the number of “items” in an array is fixed by that specified by the size of its dimension). How each "item" in the table is to be interpreted is specified by a separate data-type class, of which one is associated with every array (i.e., the “items” in an array are represented by various storage formats – ascii_integer, integer, ascii_real, real, etc).. The term record is used here to denote a data structure whose elements have heterogeneous data types.

An instance of the Table_Base class consists of a collection of contiguous one-dimensional segments of memory (owned by the table), combined with an indexing scheme that maps the "items". How many "items" in each row, how many bytes in each "item" and how the bytes are interpreted is defined by the data-type class associated with the table (i.e., basic constraints on storage order, element types, and number and length of rows are defined by the data-type class).

Example Classes:
- Binary table
- Character table

3. Unencoded Stream Base

Unencoded stream base -- describes a collection of "items" where the "items" are interpreted without any character encoding (e.g., ASCII character set).

An instance of the Unencoded_Stream_Base class consists of a contiguous stream of ASCII characters, combined with a field_delimiter scheme that maps the "items". How many "items" in each record, how the bytes are interpreted is defined by the data-type class associated with the unencoded_stream_base (i.e., basic constraints on number of fields in a record, element types, and the number of records are defined by the data-type class).

Example Classes:
- CSV_file
- Header

4. Encoded Stream Base

Encoded stream base -- describes a collection of "items" where the "items" are interpreted in accordance with a recognized International Standard (e.g., JPEG_2000).

Example Classes:
- SPICE_Kernel

4.2 PDS4 Data Product Description

TBD

5.0 PDS4 PRODUCT LABEL SCHEMA

This section introduces the concept of a product label schema and how a schema is used in the process of designing, generating, and validating the products in your archive. Objects were introduced in Section 2.1.3, object descriptions in Section 2.1.4, and products in Section 2.1.6. In Section 2.1.4.1 we explained that the practical application of object descriptions is in 'labels', which are based on XML schemata (Section 2.1.4.2). This section describes how the label for an actual product is created.

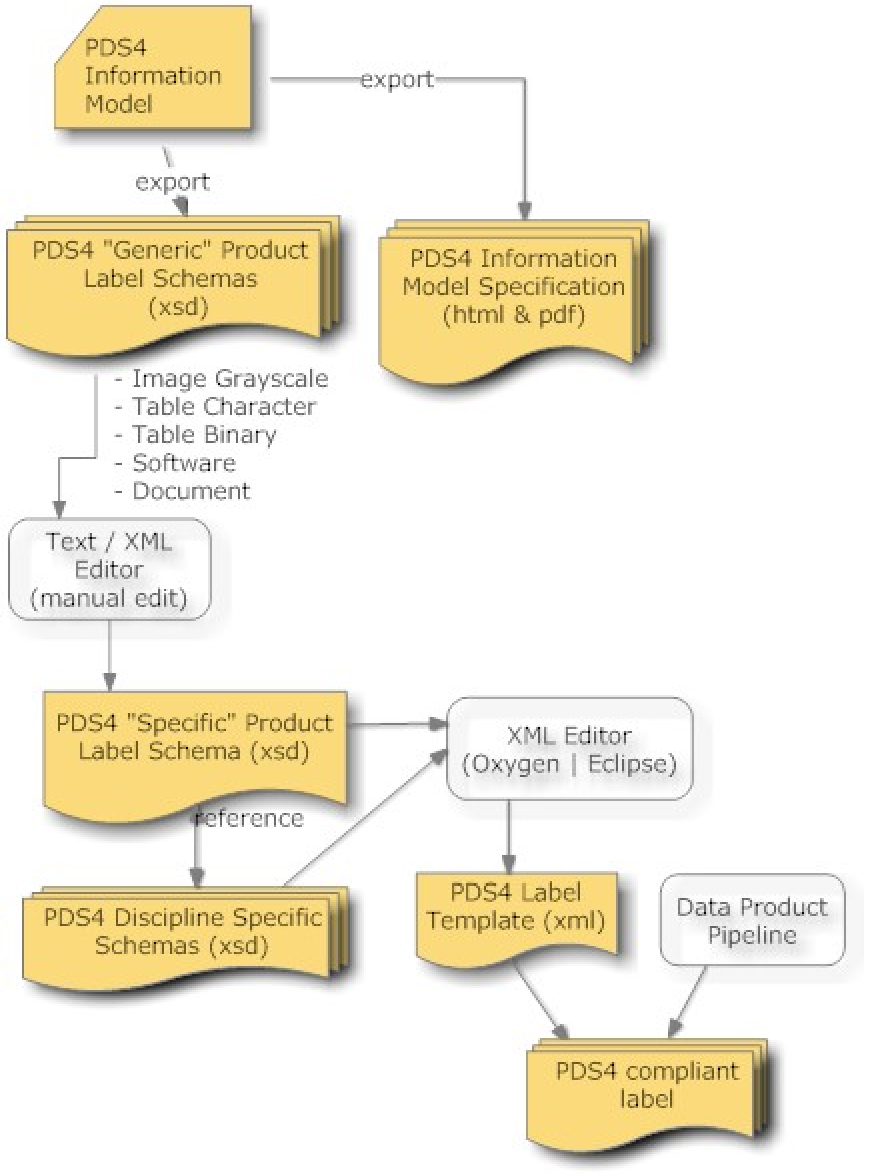

PDS4 Information Model
export
export
PDS4 "Generic" Product Label Schemas (xsd)
PDS4 Information Model Specification (html & pdf)
- Image Grayscale
- Table Character
- Table Binary
- Software
- Document
Text / XML Editor (manual edit)
PDS4 "Specific" Product Label Schema (xsd)
XML Editor (Oxygen | Eclipse)
reference
PDS4 Discipline Specific Schemas (xsd)
PDS4 Label Template (xml)
Data Product Pipeline
PDS4 compliant label

Figure 5-1. Diagram of the Lifecycle of a Product Label Schema

Figure 5-1 shows the steps between the PDS4 information model and an actual PDS4 compliant product label or set of labels. The information model includes specifications for each of the entities allowed in the archive; these can be expressed as generic XML schema documents (XSD files). The complete set of PDS4 generic schemata can be found at

http://pds.jpl.nasa.gov/schema/pds4/generic

Subdirectories hold schemata that are common across PDS as well as schemata that are defined by the PDS Discipline Nodes.

Once data products have been identified for archiving, the initial step of designing a data product should have been defined by science requirements. In most cases, the structure of the data was probably determined before your instrument was selected for the mission. The structure of the underlying data is typically obvious (e.g., table or image structure).

- TABLE - a uniform collection of ROWs and COLUMNs stored in either ASCII or binary format. ASCII forms are easily imported into a variety of spreadsheet and database applications.

- IMAGE - a two dimensional array of spatially organized measurements (LINES and SAMPLES). Many public domain image display programs can read PDS Image objects.

PDS has created sets of generic product label schemas that address all of the envisioned PDS4 structures. Your first step is to select, from the set of PDS4 "Generic" Product Label Schemas, the schema that most closely represents your data product (e.g., Image Grayscale, Table Character, Table Binary, etc).

The next step is to review the "Generic" Product Label Schema and to tailor this schema to be more specific to the product that you want to archive with the PDS. The process by which the "Generic" schema is tailored to become the "Specific" schema is, at least at this point, a manual process. Expect several iterations and use the assistance of your PDS representative.

The generic schemata incorporate many options which data providers will not be interested in adopting. An XML editor (Section 2.1.4.1.2) can quickly remove the unneeded optional sections. A simple text editor can also be employed, but it will not have the built-in error detection and verification features of an XML editor. At the same stage, special requirements imposed by the responsible PDS Discipline Node may be incorporated. The result is a 'specific schema' — another XSD file.

The "specific schema represents the overall structure and format of the archived data product; it defines, in the strictest sense, the greatest latitude permissible in validation of the product to ensure PDS compliance.

The specific schema also serves as the building block from which a label template can be derived — often ‘exported’ from an XML editor (Section 2.1.4.2.2). The label template, when used in a data processing pipeline, then allows generation of individual product labels. Whether they use the term or not, data producers will almost certainly have a need to develop a 'pipeline' for handling mission data. The pipeline begins with data collection (as from a telemetry stream) and ends with generation of standard products. Except for a few ancillary documents, the pipeline will provide most of the products you will need for your archive. PDS personnel can offer suggestions for automating the label generation process, including the use of PDS tools.

Other tools provide convenient ways to check that individual products meet PDS archiving Standards. Some validation tools can be built into the data processing pipeline. Consult your PDS rep to obtain the latest versions of validation tools and for assistance in effective use of them.

5.1 Restrictions in Tailoring Schemas

TBD

5.2 Overview of Data Dictionaries

A Data Dictionary serves several purposes. First, the dictionary serves as a reference manual to users of the PDS (and other planetary data systems) to define the attributes and classes that are used to describe planetary data and meta-data. Second, the dictionary serves as a reference for data producers (and others) to aid in the design and understanding of data descriptions. Third, the dictionary has the overall responsibility of ensuring the attributes and classes used in the data descriptions are used in a standard, consistent, predictable manner to the point where each attribute and class can be managed and used as a resource.

Conceptually, a data dictionary defines the attributes and classes which may be used in PDS4 product labels. Practically speaking, it must contain human-readable definitions as well as the syntax and semantic constraints placed on values of the attribute. For classes, it provides the explicit list of attributes constituting the class, and indicates which are required, optional, and/or repeatable. It might also indicate that one or more sub-classes are allowed (or required).

Every attribute and class that is used in any PDS label must first be defined in a data dictionary. Ultimately, all dictionaries will be integrated into the PDS4 Information Model which will “build” the PDS4 Data Dictionary document and the associated Mission and Node “dictionary” schemata.

Data dictionaries are categorized into:

- Global
- Mission specific
- Node specific

Global dictionaries are those that are used globally across all product schemata.
The global dictionaries are comprised of attributes and classes that are pervasive across a large number of the PDS product schemata.

Mission specific data dictionaries are those that are comprised of attributes and classes specific to a particular mission or investigation (i.e., the Mars Express mission could create a data dictionary for the sole purpose of defining attributes and classes that describe instrumentation and science data that is particular to the Mars Express mission). Depending upon preference, the mission may elect to have a single data dictionary (that describes both instrumentation and science data) or multiple dictionaries where one dictionary is specific to instrumentation descriptors and another is specific science data descriptors.

Node specific data dicationaries are those that are comprised of attributes and classes specific to a particular PDS node. For example, the Rings node has attributes / classes that are particular to a large number of Rings products. The Rings node may identify a number of instances where it would be convenient to group Rings descriptors into one or more data dictionaries.

5.2.1 Building and Using Local Data Dictionaries

Local data dictionaries are categorized into:

- Mission specific
- Node specific

This section describes the inter-relationships between the “generic” and “specific” data dictionary schemas, the data dictionary label(s), and the dictionary service that creates the local data dictionary schema.

The dictionary service optionally merges the attributes and classes defined in the local data dictionary with the PDS4 Information Model. This will ensure that your locally defined attributes and/or classes will be contained in the next build of the PDS4 Data Dictionary and the next build of the Generic Mission & Node schemata.

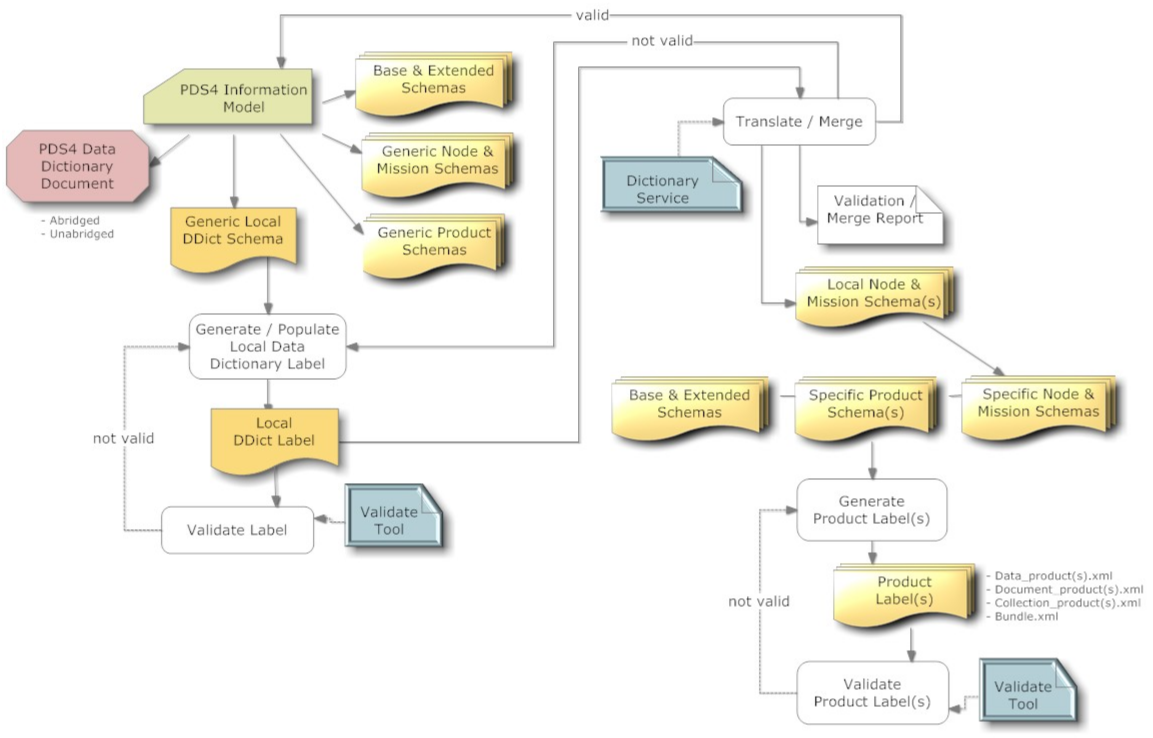

valid
not valid
PDS4 Information Model
Base & Extended Schemas
Generic Node & Mission Schemas
Generic Product Schemas
PDS4 Data Dictionary Document
- Abridged
- Unabridged
Generic Local DDict Schema
Generate / Populate Local Data Dictionary Label
Local DDict Label
not valid
Validate Label
Validate Tool
Translate / Merge
Dictionary Service
Validation / Merge Report
Local Node & Mission Schema(s)
Base & Extended Schemas
Specific Product Schema(s)
Specific Node & Mission Schemas
Generate Product Label(s)
Product Label(s)
- Data_product(s).xml
- Document_product(s).xml
- Collection_product(s).xml
- Bundle.xml
not valid
Validate Product Label(s)
Validate Tool

Figure 5-2 Local Data Dictionary Development Flow Diagram

Step #1: Select, from the set of PDS4 "Generic" Product Label Schemas, the Data-Dictionary schema, 'Local_DD.xsd'.

Note that the Data-Dictionary schema contains an embedded reference to the first global dictionary, Extended_Types.xsd, which in turn has an embedded reference to the second global dictionary, Base_Types.xsd. You will need both dictionary files to be co-located to the Local_DD.xsd in order to proceed

Step #2: Download the Data-Dictionary schema, Local_DD.xsd, and the two global dictionaries, Extended_Types.xsd and Base_Types.xsd, from:

`http://pds.jpl.nasa.gov/schema/pds4/generic/common/`

Step #3: Examine the as yet unmodified "Generic" Data-Dictionary schema in your favorite XML editor (e.g., Oxygen or Eclipse). You may also examine the schema in a text editor (e.g., UltraEdit, BBEdit, etc). Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML).

Note that if there are errors in the XML schema, contact your PDS representative for further instructions on how to resolve any discrepancies.

In this case, the “Generic” schema represents the “Specific” overall structure and format of the data dictionary product. The “Generic” schema is immutable in that it defines, in the strictest sense, the greatest latitude permissible in the validation of the data dictionary product label to ensure PDS compliance (i.e., don’t even attempt to modify the “Generic” schema – everything is there that is needed to create new attributes / classes).

Step #4: Using your favorite XML editor (e.g., Oxygen or Eclipse), generate a local data dictionary XML label from the Data-Dictionary schema. Most XML editors provide a capability to “export/create” an XML label from an XSD. You will want to use this feature to export/create a sample label (which is an XML file) from the Data-Dictionary schema (which is an XSD file). Save the sample (as-yet-unedited) data dictionary label.

Step #5: Examine the as-yet-unedited data dictionary label in either your favorite XML editor or text editor. Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML. As the sample label was generated by the XML editor, there shouldn’t be any errors. Contact your PDS rep to resolve any discrepancies.

Step #6: Using the XML or text editor, enter / populate the data dictionary label with the metadata associated with each attribute and class that is to be defined in the local data dictionary. Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML).

You can use the PDS4 Validation Tool to further validate the contents of the XML label. The Validation Tool and associated documentation can be found online at:

ftp://pds.nasa.gov/pub/toplevel/2010/preparation/validate-0.1.0-bin.zip

Step #7: At this point, you should have a fully formed compliant XML data dictionary label. A decision point has been reached whereby the contents of the locally defined data dictionary may be “merged” with the PDS4 Data Dictionary (via ingestion into the PDS4 Information Model).

To “merge” the locally defined data dictionary with the PDS4 Data Dictionary, set the “PDS4_merge_flag” to “true”. If you choose not to “merge”, then set the “PDS4_merge_flag” to “false”.

Step #8: Access the online Data Dictionary Service, “point” the service to your data dictionary label, and press “go”. The Data Dictionary Service will process your file and create a local instance of the “Node & Mission” schemata (XSD) and a report describing any anomalies encountered while translating / merging.

If you have elected to “merge” the locally defined data dictionary with the PDS4 Data Dictionary, the service will additionally validate the contents to ensure a compliant “merge” of the elements and classes is possible (e.g., no duplication of element / class names within the same Registration Authority, etc.). If the Service doesn’t detect any anomalies, then the Service will “register” the contents of your data dictionary label with the PDS4 Information Model. This will ensure that your locally defined attributes and/or classes will be contained in the next build of the PDS4 Data Dictionary.

The Data Dictionary Service can be accessed online at:

http://pds.jpl.nasa.gov/tbd/...

Step #9: Now that you have a “valid” local instance of the “Node & Mission” schemata (XSD), you can incorporate these dictionaries into your data product pipeline. This is done through an XML include reference in your specific product schema. So, the next step is to “link” the set of “Node & Mission” schemata (that are to be referenced) into the product schema.

Using the XML or text editor, add the equivalent of the following XML statements to the product schema.

```
  <xsd:include schemaLocation="Extended_Types_edited_0111c.xsd">
    <xsd:annotation>
      <xsd:documentation>PDS (common) Data Dictionary</xsd:documentation>
    </xsd:annotation>
  </xsd:include>

  <xsd:include schemaLocation="MarsExpress_instrumentation.xsd">
    <xsd:annotation>
      <xsd:documentation>
         Mars Express Data Dictionary for instrumentation
      </xsd:documentation>
    </xsd:annotation>
  </xsd:include>

 <xsd:include schemaLocation="Rings_Prod_Info.xsd">
    <xsd:annotation>
      <xsd:documentation>
         Rings Node Data Dictionary
      </xsd:documentation>
    </xsd:annotation>
  </xsd:include>
```

In the above, the first <xsd:include> instantiates a reference to the first global dictionary, Extended_Types.xsd. Recall, that Extended_Types.xsd has an embedded reference to the second global dictionary, Base_Types.xsd. The second <xsd:include> instantiates a reference to the “Mission” schema (that we created), "MarsExpress_instrumentation.xsd.

The third <xsd:include> instantiates a reference to the “Node” schema (that we created), "Rings_Prod_Info.xsd.

Step #10: Now that you have all of the pieces in place, you can incorporate all of these files into your data product pipeline that will pump out gazillions of PDS compliant labels --- don’t forget to validate, and then validate again, and again, and again…

Step #11: Repeat the above process for all the products in your archive. This includes most (but probably not all) of the following:

- bundle.xml
- collection_misc.xml
- aareadme.xml
- errata.xml
- collection_about.xml
- about_product(s).xml
- collection_browse.xml
- browse_product(s).xml
- collection_calibration.xml
- calibration_product(s).xml
- collection_context.xml
- context_product(s).xml
- collection_data.xml
- data_product(s).xml
- collection_document.xml
- document_product(s).xml
- collection_gazetter.xml
- gazetter_product(s).xml
- collection_geometry.xml
- geometry_product(s).xml
- collection_SPICE.xml
- spice_product(s).xml
- collection_xml_schema.xml
- xml_schema_product(s).xml

Step #12: The next step in the process, is to register all of the products in your archive. See Section x.x.x for a description of the registration process.

5.3 Example of Relationship of Schemas to Labels

This section illustrates the lifecycle process of the “generic” and “specific” product label schemas and how they relate to the label template and the resulting product labels. The above is demonstrated by using an example PDS3 data product.

The example product is a simple ASCII table that is currently in the PDS3 archive.

- MGS-M-RSS-5-TPS-V1.0: A radio science data set that seems to consist of well-behaved ASCII tables with little or no additional keywords beyond those in a basic label. There are two tables in each label, but both tables are in the same file (one is a single line of header parameters).

The files that describe both the PDS3 and the PDS4 data products can be found at:

- PDS3 ODL Label: http://tbd
- PDS3 data product: http://tbd

- PDS4 XML Label: : http://tbd
- PDS4 XML Label template: : http://tbd
- PDS4 Generic Schema: : http://tbd
- PDS4 Specific Product Schema: : http://tbd
- PDS4 Specific Data_Set Schema: : http://tbd

Note that at this time, the above examples are out of date with the current information model and therefore do not provide an exact representation of the current schemas.

Step #1: Select, from the set of PDS4 “Generic” Product Label Schemas, the schema that most closely represents your data product (e.g., Image Grayscale, Table Character, Table Binary, etc)

Step #2: Download the “Generic” Product Label Schema from:

`http://pds.jpl.nasa.gov/schema/pds4/common/`

Step #3: Make a copy of the “Generic” Product Label Schema and save the copy as the “Specific” Product Label Schema.

Step #4: Examine the as yet unmodified “Specific” Product Label Schema in your favorite XML editor (e.g., Oxygen or Eclipse). You may also examine the schema in a text editor (e.g., UltraEdit, BBEdit, etc). Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML).

Note that if there are errors in the XML schema contact your PDS representative for further instructions on how to resolve any discrepancies.

Step #5: Use the editor to tailor this schema to be more specific to the product that you want to archive with the PDS. The "Specific" schema represents the overall structure and format of the archived data product. The "Specific" schema defines, in the strictest sense, the greatest latitude permissible in the validation of the product label to ensure PDS compliance.

Examples of types of "edits/restrictions" that might be appropriate with respect to the specific schema; include:

1) Restrict the set of all possible target names to a single value (e.g., MARS).
2) Restrict the instances in the File_Area_Type to a single reference to the type of file being described (i.e., in our example we are describing a character table having fixed length records – so we would remove all instances except the reference to File_Character_Fixed).
3) As our example table product does not have any "Statistics", remove all references to Object_Statistics_Type
4) As our example table product does not have any "Special Constants", remove all references to Special_Constants_Type.

Expect several iterations and use the assistance of your PDS representative.

Step #6: Save the edited/tailored "Specific" Product Label Schema.

Step #7: Most XML Editors provide a capability to "export/create" an XML label from an XSD. You will want to use this feature to export/create a sample label (which is an XML file) from the "Specific" schema (which is an XSD file). Save the sample label.

Step #8: Examine the sample label in either your favorite XML editor or text editor. Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML. As the sample label was generated by the XML editor, there shouldn't be any errors. Contact your PDS rep to resolve any discrepancies.

Step #9: Now that you have a "valid" XSD and sample label, we can proceed with creating a data product pipeline that will pump out gazillions of PDS compliant labels.

Validating the data product labels is where the data product schemas become invaluable. The use of XML in data product labels and in schemas provides an expedient method by which your pipeline can ensure your product labels are PDS compliant. The PDS can offer suggestions for automating the validation process; including, the use of PDS tools.

5.4 Validating the Relationship of Schemas and Labels

This section describes the process of validating the object-oriented design and the inherent relationships of and between the generic schema, the specific schema, and the resulting child XML document,

Figure 5-3 illustrates the process by which users ensure the resulting XML documents are compliant to the parent schemas. The validation process guarantees the object-oriented design of the parent-child relationships are preserved through out the design and implementation stages of preparing XML documents; specifically that:

1. The "Specific" Product schema validates/are valid against the "Generic" Product schema.

2. The "Label Template" validates/are valid against the "Specific" and the "Discipline Specific" schemas.

3. The PDS4 complaint labels validate/are valid against the "Specific" and the "Discipline Specific" schemas.

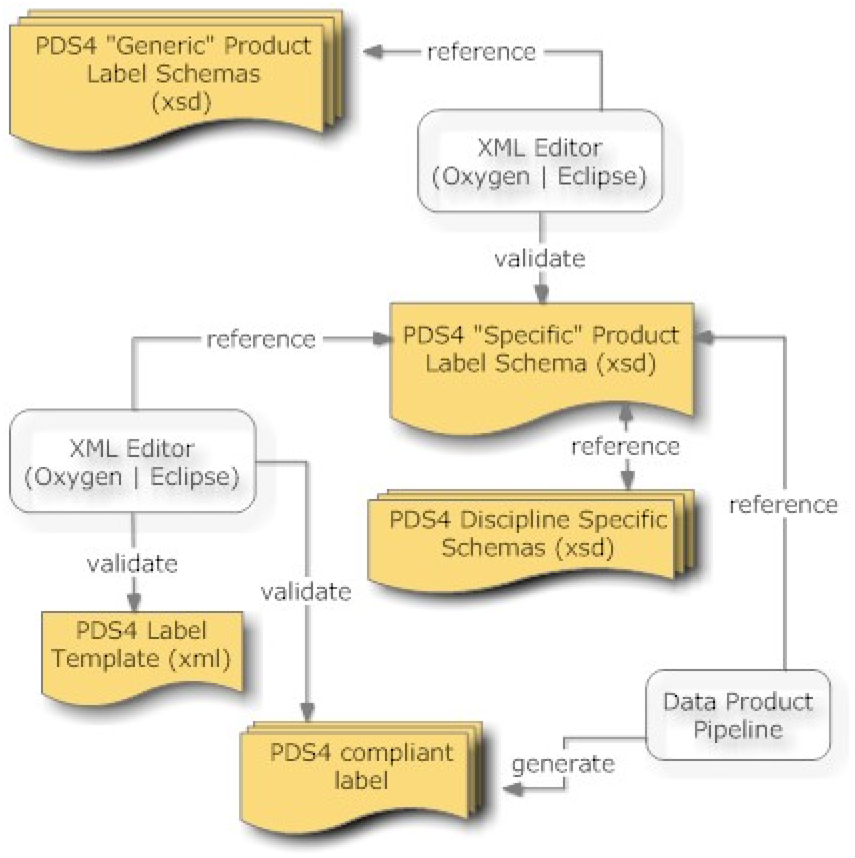

PDS4 "Generic" Product Label Schemas (xsd)
reference
XML Editor (Oxygen | Eclipse)
validate
reference
PDS4 "Specific" Product Label Schema (xsd)
reference
XML Editor (Oxygen | Eclipse)
PDS4 Discipline Specific Schemas (xsd)
reference
validate
validate
PDS4 Label Template (xml)
Data Product Pipeline
PDS4 compliant label
generate

Figure 5-3. Diagram of the Validation Lifecycle of a Product Label Schema

The underlying mechanism by which the above three steps is accomplished is noted as an XML editor (e.g., Oxygen or Eclipse). But, there are alternate mechanisms which could be used in place of your favorite XML editor; such as, an XML/XSD aware application. A machine-assisted mechanism for ensuring the "Specific" schema is valid against the "Generic" schema has yet to be determined.

6.0 PDS4 DATA PRODUCT GENERATION

This section introduces the concept of …

TBD

7.0 PDS4 DATA PRODUCT TRANSFER

This section describes the process of transferring products within and external to the PDS.

7.1 Product Transfers

Transfers of products from point to point within and external to the PDS can include packaging mechanisms like ZIP, gzip, tar, physical media such as thumb drives, external hard drives, etc. The actual transfer mechanism should be coordinated with the node with which you are working.

Step #1: Select, from the set of PDS4 "Generic" Product Label Schemas, the Product Zipped schema, 'Product_Zipped.xsd'.

Step #2: Download the Product_Zipped schema, and the two global dictionaries, Extended_Types.xsd and Base_Types.xsd, from:

http://pds.jpl.nasa.gov/schema/pds4/generic/common/

Step #3: Make a copy of the "Generic" schema and save the copy as the "Specific" Product Label Schema.

Step #4: Examine the as yet unmodified "Specific" schema in your favorite XML editor (e.g., Oxygen or Eclipse). You may also examine the schema in a text editor (e.g., UltraEdit, BBEdit, etc). Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML).

Note that if there are errors in the XML schema contact your PDS representative for further instructions on how to resolve any discrepancies.

Step #5: Use the editor to tailor this schema to be more specific to the product that you want to use. The "Specific" schema represents the overall structure and format of the product. The "Specific" schema defines, in the strictest sense, the greatest latitude permissible in the validation of the product label to ensure PDS compliance.

Examples of types of "edits/restrictions" that might be appropriate with respect to the specific schema; include:

1) Restrict the set of values in the <Subject_Area>
2) Restrict the instances in the File_Area_Type to a single reference to the type of file being described (i.e., in our example we are describing a character table

having fixed length records – so we would remove all instances except the reference to File_Character_Fixed).

Expect several iterations and use the assistance of your PDS representative.

Step #6: Save the edited/tailored "Specific" Product_Zipped schema.

Step #7: Most XML Editors provide a capability to "export/create" an XML label from an XSD. You will want to use this feature to export/create a sample label (which is an XML file) from the "Specific" schema (which is an XSD file). Save the sample label.

Step #8: Examine the sample label in either your favorite XML editor or text editor. Ensure that the XML is fully formed (i.e., the XML editor will validate the XML and will have an indicator (which is usually a green or red box) that indicates if errors are present in the XML. As the sample label was generated by the XML editor, there shouldn't be any errors. Contact your PDS rep to resolve any discrepancies.

Step #9: Now that you have a "valid" XSD and sample label, we can proceed with populating the label with the required and optional information that will define the set of products being transferred.

Validating the data product labels is where the data product schemas become invaluable. The use of XML in data product labels and in schemas provides an expedient method by which you can ensure your product labels are PDS compliant. The PDS can offer suggestions for automating the validation process; including, the use of PDS tools.

Step #10: In the <FILE_AREA>, point to the zip file. Use the <Zipped_Member_Entry> to reference each product being zipped.

Step #11: Now that you have the zipped file and the Product_Zipped.XML label that describes the contents of the zip file, you are ready to copy these to physical media such as thumb drives, external hard drives, etc.

Step #12: Negotiate the actual transfer with the node – single transfer of the entire archive or incremental deliveries.

7.2 Delivery of Accumulating Archives

TBD

8.0 EXAMPLE PDS4 PRODUCTS

A PDS4 tutorial would not be complete without providing a set of PDS4 products that were generated from example PDS3 products.

There are two different sets of examples:

- The first is a set of example products. This includes a representative set of products that would exist within an archive (e.g., character table, binary table, document, etc.).

- The second set is a complete archive. This includes all products that would comprise a complete PDS4 archive (e.g., bundle, collections, collection inventories, aareadme, errata, and products). The archive is presented in a directory structure required of an archive.

8.1 PDS4 Product Examples

The set of PDS4 product examples can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/dph_examples/

The HTML page that provides detailed descriptions of the PDS4 products can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/dph_examples/PDS4ExampleDataProductClasses.htm

Within the set of examples, there is an example of the following product types:

1. the IMAGE_GRAYSCALE extension of the PDS4 Array_Base, (i.e., Homogeneous N-dimensional array of Scalars) class where a contiguous stream of BINARY data, assembled as a two dimensional data structure, maps the "items" contained in a IMAGE_GRAYSCALE file.

2. the TABLE_CHARACTER extension of the PDS4 Table_Base (i.e., Heterogeneous repeating record of Scalars) class where a contiguous stream of ASCII characters, assembled as fixed-width fields, maps the "items" contained in a TABLE_CHARACTER file.

3. the TABLE_BINARY extension of the PDS4 Table_Base (i.e., Heterogeneous repeating record of Scalars) class where a contiguous stream of BINARY data, assembled as fixed-width fields, maps the "items" contained in a TABLE_BINARY file.

4. the TABLE_CHARACTER_GROUPED extension of the PDS4 Table_Base (i.e., Heterogeneous repeating record of Scalars) class where a contiguous stream of ASCII characters, assembled as sets of repeating fixed-width fields, maps the "items" contained in a TABLE_CHARACTER_GROUPED file.

5. the STREAM_DELIMITED class where a contiguous stream of ASCII characters, combined with a field_delimiter and record_delimiter scheme, maps the "items" contained in a CSV “like” file.

6. the DOCUMENT_SET class where one or more instantiations of a document (e.g., ascii text, pdf, html), as identified as a set, comprise a logically complete “copy” of the referenced document product.

8.2 PDS4 Example Archive

This set of PDS4 examples were derived from a PDS3 PPI data_set:

```
    DATA_SET_ID = "VG2-J-PLS-5-SUMM-ELE-MOM-96.0SEC-V1.0"
```

Both the original PDS3 data_set files and the equivalent PDS4 product files are presented for the purposes of illustrating (comparatively) how a PDS3 data_set can be migrated to PDS4.

8.2.1 PDS3 Data_set Files

The directory structure of the original PDS3 PPI data_set consisted of the following:

```
- aareadme.txt
- checksums.txt
- errata.txt
- voldesc.cat
- \browse
- \catalog
- \data
- \documents
- \documents\mission
```

- `\documents\pls`
- `\documents\symbols`
- `\index`

The following PDS3 files have no PDS4 equivalents and therefore were not migrated:

- `voldesc.cat`
- `\index`

The original PDS3 PPI data_set can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/dph_example_PDS3_VG2PLS/

A zip file of the PDS3 VG2PLS data_set can be downloaded from:

http://pds.jpl.nasa.gov/repository/pds4/examples/dph_example_PDS3_VG2PLS/VG2PLS.zip

8.2.2 PDS4 Product Files

The PDS4 archive structure, as migrated from the above PDS3 files, consists of the following:

- `aareadme.txt`
- `checksums.txt`
- `errata.txt`
- `Product_Text_File_aareadme_20101129.xml`
- `Product_Text_File_checksums_20101129.xml`
- `Product_Text_File_errata_20101129.xml`
- `collection_miscellaneous_root_20101129.xml`
- `manifest_root_collection_20101129.tab`
- `Product_Bundle_20101129.xml`
- `\browse`
- `\context`
- `\data`
- `\documents`
- `\schemas`

The PDS4 product files can be found at:

http://pds.jpl.nasa.gov/repository/pds4/examples/dph_example_archive_VG2PLS

For more information, contact…

APPENDIX A ACRONYMS

The following acronyms are pertain to this document:

ADM	Architecture Development Method
API	Application Programming Interface
COTS	Commercial Off-The-Shelf
EN	Engineering Node (PDS)
ESDIS	Earth Science Data and Information System
FTP	File Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IPDA	International Planetary Data Alliance
IT	Information Technology
JPL	Jet Propulsion Laboratory
NASA	National Aeronautics and Space Administration
NSSDC	National Space Science Data Center
PDS	Planetary Data System
RM-ODP	Reference Model of Open Distributed Processing
RSS	Really Simple Syndication
SDSC	San Diego Supercomputing Center
SOA	Service-Oriented Architecture
TB	Terabyte
TOGAF	The Open Group Architecture Framework
XML	eXtensible Markup Language

APPENDIX B DEFINITION OF TERMS

The following are definitions of essential terms used throughout this document:

Association:
An "association" is a type of defined relationship between classes.

Attribute:
An "attribute" is a property or characteristic that allows both identification and distinction.

Cardinality:
"Cardinality" is 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 attribute that may be multi-valued will have a cardinality of "1..*". 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. At least one value is always required in PDS4.

Class:
A "class" is the set of attributes which identifies a family. A class is generic -- a template from which individual members of each family may be constructed.

Class Hierarchy:
A "class hierarchy" is a classification of object types, denoting objects as the instantiations of classes.

Data Elements:
A "data element" is a discrete unit of data or metadata. It is an elementary piece of information in a data dictionary.

Entity:
An "entity" is something that has a distinct, separate existence.

Metadata:
Metadata is data about data.

Model:
A "model" is a representation or description designed to show an entity and its composition.

Object:

An "object" is a specific instance of a class.

Use to describe in cross_reference_area

Product Type	*reference_association_type*	*referenced_object_type*
Product_Browse	has_browse	Product_Browse
Product_Calibration	has_calibration	Product_Calibration
Product_Geometry	has_geometry	Product_Geometry
Product_PDS_Affiliate or Product_PDS_Guest	has_personnel	Product_PDS_Affiliate or Product_PDS_Guest
Product_SPICE	has_spice	Product_SPICE
Product_Target	has_target	Product_Target
Product_Thumbnail	has_thumbnail	Product_Thumbnail
Product_Document	has_document	Product_Document
Product_Investigation	has_investigation	Product_Investigation
Product_Instrument_Host	has_instrument_host	Product_Instrument_Host
Product_Instrument	has_instrument	Product_Instrument
Product_Node	has_node	Product_Node
Product_Resource	has_resource	Product_Resource