PDS4 Data Standards Concepts

1. Preface

This document provides overviews and some basic terminology useful for understanding and applying the PDS4 data formatting standards.

1.1 Audience

This manual is intended for a general audience with little or no background in PDS, XML or data preparation. It is appropriate for both data preparers and end users who wish to familiarize themselves with basic PDS requirements, structures and definitions before diving into the full text of the data standards or tool documentation.

1.2 Why have data standards?

Standards lie at the heart of information transfer and interoperability. It is standards that allow us to browse the web, make online purchases, and even open and print this document. PDS standards provide a consistent framework for describing planetary science data, so that data providers, programmers and end-users all know what to expect when creating and working with PDS files. Standards also help the PDS ensure that data that is collected and archived today will still be readable and usable generations from now, which in turn ensures a return on the investments made in flying planetary space missions and funding ground-based planetary research.

1.3 A Quick Introduction to the Planetary Data System

NASA's Planetary Data System (PDS) is a federation of *nodes*, each of which focuses on a specific planetary science discipline. When a data preparer first contacts the PDS, one of these discipline nodes will be assigned as the *principal node* or *consulting node* for the mission, instrument, or project. A large mission may have different consulting nodes for different groups of instruments, and when this happens one discipline node will be designated as the *lead node* for PDS.

The consulting node provides basic training, copies of standards documents and tutorials, development materials and advice throughout the archive development effort. As the mission or project archive is being designed, the PDS consulting node will provide details of what additional information, over and above the observational results, will be required for inclusion in the archive. When data production starts, the consulting node will also provide validation support for standards compliance, conduct an external peer review of data submitted and continue to provide validation support and advice through final delivery of the edited data.

More details about the individual discipline nodes, the peer review process, and available tools can be found **[TBD: This might be a single reference point, a URL, a list of specific titles, etc. It will be added when we have a better idea what the**

document and tutorial set will look like for the first public release.]

1.4 Related Documentation

[This will be added for the second draft.]

1.5 Conventions

[This will be added for the second draft.]

I. What is PDS4?

This section provides an overview of the PDS4 data formatting standards and the terminology used in the process of designing labels and preparing data for archiving. The process of organizing observational data and documents into sequences of bytes for archiving is actually fairly simple and straight-forward. The process of documenting the format, circumstances and context of those data, however, is fairly involved and is the job of the PDS4 *product label*.

2. Overview of Data Structures

PDS4 data structures are fairly tightly constrained, particularly for observational data, for which PDS has defined its own archival formats. Where PDS does not define its own particular format, it has designated specific external standards – and in many cases specific subsets of those standards – as acceptable for archival information.

2.1 Observational Data

The vast majority of observational data can be broken down and formatted into a series of one or more of two basic formats:

1. N-dimensional arrays
2. Fixed-width tables

Images, for example, are either 2- or 3-dimensional arrays of pixels all of the same type, while fixed-width tables (text or binary) are used for storing data contained in records with fields of potentially different types. The PDS4 data standards also dictate what storage formats are acceptable for pixels in arrays and fields in tables.

2.2 Documents

Documentation includes any textual or text-based information supplied with the data to assist the user in understanding, interpreting, calibrating, or otherwise manipulating the data. Frequently such documentation includes graphics or images to assist in comprehension. PDS has adopted the PDF/A standard as the archival format for complex documents. UTF-8 encoded text files may be used for simple documents that do not require graphics or special formatting control.

2.3 Supplementary Information

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

- Browse facilities;

- Descriptions of spacecraft, instruments, observing conditions, etc.;
- Calibration data, including observations as well as things like flat-fields, filter curves, transformation tables, etc.;
- Observing and command logs;
- Alternate formats for archival documents;

And so on. Some of these things, like instrument descriptions, are essential to the long-term archive and are required to be present. Data providers are also encouraged to include supplementary information wherever it might be helpful to an end user.

Data providers will be given a list of required supplementary data as part of the initial design process with their consulting PDS node.

3. Overview of XML and XML Schema

The current implementation of the PDS system uses XML for all labels and for inter-process communication. XML Schema is used to define details of label structure and to constrain label content as part of quality assurance and validation for all labels.

3.1 What is XML?

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

3.2 Why XML?

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

In addition, the namespace aspect of XML (and XML Schema, following) provides a convenient method for implementing *local data dictionaries*, as described in the next chapter. Local dictionaries will contain new keywords defined by the data preparer as needed to contain detailed descriptions of the various products being created. They are an essential tool for allowing data preparers latitude to tailor their labels to the data in hand without impacting the more general, required structures in the labels.

3.3 What is XML Schema?

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

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

3.4 Why XML Schema?

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

As with XML, there are many third-party tools that support the creation and application of XML Schema files, and the schema files created can be used to validate and verify the contents of the corresponding XML labels. Further, schema files contain detailed documentation of label structure and contents, which can be saved as part of the supplementary information for the data in the archive.

In fact, PDS uses only a subset of the capabilities available in XML Schema, and has developed procedures and tools to assist users in working with XML Schema to produce the schemas needed for their XML labels.

3.5 Terminology

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

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

In this example, anything inside "<>" is called a *tag*. So `<movie>`, `<title>`, and `<storyBy>` are all tags. The corresponding closing tag, which is required, begins with "</". The string between the opening and closing tag is called the *content*. The opening and closing tags and content together are called an *element*. So "`Bedtime for Bonzo`" is the content of the `title` element, and the content of the `movie` element

is all the elements (the tags with the all their content) that come between `<movie>` and `</movie>`.

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

Note that XML uses the term "attribute" to indicate a qualifier on an element. For example, in

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

XML refers to `unit` *as an* attribute *of* `length`. *PDS only uses XML attributes in a few very specific cases. The terms "XML attribute" or "tag attribute" will always be used to indicate the case illustrated above, in order to distinguish it from the common PDS use of "attribute".*

3.6 What is a Namespace?

In general, a *namespace* is a context for defining something. For example, in the phrases "car title" and "movie title", the word *title* has two very different meanings. "Car" and "movie" provide a context that has a significant effect on how you interpret the meaning of "title" in each case.

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

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

The namespaces are `movie` and `car`, respectively. The value of `<movie:title>` will no doubt be a string, but the value of `<car:title>` is likely to be a series of sub-elements containing information like the VIN, make, model, year, registered owner, and so on. Each XML namespace is defined by a separate XML schema document.

Some additional detail: The `movie:` and `car:` prefixes in this example are typical of how namespaces are referenced in an actual XML file, but in practice these strings would actually be nicknames (abbreviations, if you prefer) for the true namespace identifiers, which most often look like URLs. The association between the nicknames and the actual namespace identifiers is made at the top of the XML file. (See section 12.1.3 Using namespaces in an XML file" for additional detail and examples.) The specific nicknames used aren't significant to XML, but choosing reasonable nicknames and using them consistently is very significant to human readers. In PDS, XML label files will not only define these prefixes, but also provide explicit pointers to the XML Schema files containing the element definitions for

each namespace, so that the label can be validated against the definitions in each dictionary.

In PDS4 labels, XML namespaces are used to associate elements with their source data dictionaries (see "4. Overview of Data Dictionaries"). All the elements from a particular namespace will be defined in a corresponding dictionary schema file. There will be a global namespace for the PDS elements common to all labels; node namespaces to contain the elements specific to node disciplines; and each data preparer may create a dictionary to define elements in a local namespace for inclusion in his product labels.

4. Overview of Data Dictionaries

Data dictionaries define the meaning and structure of terms used in PDS labels as well as constraining their content. The attributes and classes (see "3.5 Terminology") used in any PDS label must first be defined in a data dictionary file.

4.1 What is a Data Dictionary?

In general, a *data dictionary* contains a list of terms and definitions used to describe some collection of data. In the case of the PDS, the data dictionary defines attributes and classes used in PDS XML files. A data dictionary will define the concepts embodied by each attribute ("start time", "instrument name", etc.) as well as defining and constraining the values which may appear as content between the corresponding XML tags.

For example, consider our `movie` class from "3.5 Terminology". The schema that defines the structure of the `<movie>` element must also define all the elements it contains, like `<title>` and `<starring>`. Most of these will be defined as simple text strings, but `<firstRelease>` will be defined in the schema as an integer, and can be constrained to have a value of 1878 or greater.

In PDS4, a data dictionary is expressed as an XML Schema file; each data dictionary schema file defines the elements corresponding to a single namespace.

4.2 PDS4 Data Dictionaries

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).

Physically, any data dictionary will exist in several forms:

- As an XML file created by a data preparer ***[Note: This is an implementation assumption that may change.]***
- As an XML Schema file used for validation of labels
- As entries in the PDS Dictionary Data Base

The XML form will be created by data preparers for input to utilities that will create other forms, like the XML Schema form, the input for the PDS Dictionary Data Base (DDB), or a human-readable document for reviewers to reference when evaluating the data in peer review. ***[Implementation assumption]*** Ultimately, all dictionaries will be loaded into the PDS DDB, which integrates all dictionaries into a single master data base that users may query.

4.3 Creating Data Dictionaries

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

Data dictionaries are created by PDS node personnel and by data preparers. Node personnel compile dictionaries that include terms common to the node discipline, so that key concepts and parameters can be defined uniformly across data sets from multiple sources. A data preparer creates a *local dictionary* when he wishes to define attributes and classes specific to his mission, observing campaign, data restoration effort, etc.

There are two approaches to creating a dictionary file for PDS archive use:

1. Use the dictionary creation kit available from PDS; or
2. Create your dictionary directly in the PDS Data Dictionary Schema format.

The dictionary creation kit contains a dictionary creation schema, PDS schemas containing type parameters and unit lists, a utility for converting a dictionary XML file into the schema files needed for creating labels and submitting the dictionary to the PDS DDB. This kit should be able to walk you through creating attributes and classes for all your product labels even if you have no previous experience with XML Schema.

If you as a data preparer want finer control or want to take advantage of more advanced XML Schema features in the labels you are creating (staying within the bounds of what PDS allows, of course), you may wish to create your schema directly following the PDS data dictionary template. The PDS Data Dictionary Schema contains all the information needed to load new attributes and classes into the PDS DDB. Once your local dictionary is loaded into the DDB, you can use the DDB service to download versions suitable for validating product labels and peer-reviewing definitions.

4.4 Using Data Dictionaries

The data dictionary schema file is directly referenced by all XML label files that use attributes defined in that dictionary. The typical PDS product label will reference at least three different dictionary files:

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

Referencing a data dictionary schema generally involves two steps: First, a prefix is defined as shorthand for the namespace assigned to the dictionary; second, a reference is provided to point to the physical file containing the dictionary schema, so that validation utilities can find and read the file and apply its constraints to the label or schema being created.

The actual mechanics of how this is done varies slightly depending on whether you are referencing the dictionary schema from an XML file or an XML Schema file. ***[Need a reference to explicit examples, in this document or elsewhere.]***

4.5 The PDS Dictionary Data Base

This section can't be filled in until the Dictionary Data Base interfaces are defined.

5. Overview of Product Labels

PDS4 product labels are XML files that describe the contents of one or more data files (observational data, document files, calibration tables, etc.). The PDS label will always be in a separate file from the data/document it describes. The PDS label in conjunction with the data file(s) it describes is referred to as a *product*.

PDS labels will always be based on a *product schema*, often one tailored specifically to the data in question by the data preparer. Product schema files define the detailed structure of data/document files, and include attributes and classes from the global, node and local data dictionaries, as needed. Product schemas are usually created by the data preparer, based on a template provided by the consulting PDS node.

5.1 Content

PDS labels serve several major purposes:

1. They provide the unique identification for each product in the PDS;
2. They contain a pointer to the physical file(s) containing the data or documentation;
3. They describe the data structure of the various pieces of the product, whether it is an observational result, a document, or supplementary information; and

4. They contain detailed documentation for the product, such as observing conditions or processing status for observational data; authorship, abstract and copyright information for documents; and so on.

PDS will define the classes in the global data dictionary that are common across labels from many different sources, as well as certain classes required to support basic search, retrieval and display functions that vary depending on the type of information being labeled. The PDS nodes will define specialist classes in the node data dictionaries to contain information relevant to their specific user communities, like map projection information for certain images. Finally, the data preparer can define additional classes in a local data dictionary specific to the mission or data he is preparing to hold the various bits of documentation and history information that need to stay with the data.

5.2 Organization

PDS labels are XML documents, and will always contain a single root tag. Inside this tag, PDS organizes information into *classes* of *attributes*. Each attribute is expressed as a single XML tag with a simple string as its only content. Classes are tags which contain groups of attributes. Within the label, all attributes must be part of a class. Some classes may contain sub-classes (i.e., classes may be nested).

The classes required by PDS will be placed at the top of the label, starting with the identification class required in all PDS labels. This class defines the unique identifier for the label and its contents, as well as versioning information.

Node classes and classes created by the data preparer may be inserted at specific points, as allowed by the governing XML Schema.

5.3 Design

Most labels will first be designed in XML Schema, via a multi-stage process:

1. *Generic schemas* covering a variety of general data types – observational data, documents, browse facilities, etc. – are created and maintained by the PDS as a whole and are available from a central location.
2. Starting with one of these generic schemas downloaded from PDS, the node working closest with the data provider will remove extraneous classes not needed for the data in question, add classes needed by the node constituency, and query the other nodes for additional classes needed by their constituencies. The result is a *tailored schema*.
3. This tailored schema is then passed to the data preparer, who may insert additional classes specific to his mission, instrument, observing campaign, etc., at specific points in the

tailored schema. The result is the final, *specific schema* for the data being labeled.

The specific schema can then be used in creating and validating labels for the individual products.

5.4 Creation

Most labels in the PDS will be created using one of two methods:

1. A program will fill in a label template with specific values for each product – the most likely case for data written by a pipeline; or
2. A user will create a label in an editor – the most likely case for one-off labels like those used for documents and small supplementary products.

Both of these methods will likely involve an *XML-aware editor* at some point. An XML-aware editor can help enforce the syntax rules of XML, and in many cases such an editor can use an XML Schema file to create the basic layout with XML tags, provide an easy interface to guide a user through entering valid values, and validate the resulting XML document against the schema file.

So in the first case above, an XML-aware editor can create the label skeleton and the user would enter markers for values to be supplied by the pipeline program. In the second case, the user can create and validate a one-off label entirely within the editor.

5.5 Validation

Finished labels must be validated against the specific schema, tailored schema and generic schema on which they are based. This validation can be done inside an XML-aware editor, through programming libraries in languages such as Perl and Java, or with command-line utilities such as the *Linux xmllint* utility. Data preparers should learn early on how to validate their labels using their locally available software, because labels that fail validation will be rejected by the PDS node receiving them. Your consulting PDS node can help you get started with this, and provide you with sample labels and schemas that should pass validation.

6. Overview of Archive Organization

The word *archive* is used generally to reference large collections of data. When referring to the entirety of the PDS holdings, it is usually capitalized. In this document, we will use *archive* to mean "the entire collection of data, documents, and supplementary information created by a data preparer." So, for example, when we talk about a *mission archive*, we mean all the files coming from a single mission.

Because the typical archive usually contains thousands, if not hundreds of thousands of files, some organization is required just to manage simple bookkeeping and tracking. This section describes the hierarchy used to physically and logically organize archives created by data preparers.

6.1 Products

A PDS *product* is an XML label file plus all the data files it references. (Here we are using the term *data file* generically to indicate a file containing the observational data, document or supplementary information the label is describing, but *not* the schema files that the XML label references for its definition and validation.) A single product label may reference more than one data file, but a data file may be referenced by only one label. The data files referenced by a product label must either be in the same directory as the label, or in a subdirectory of the directory containing the label. Every product label contains a logical identifier for the product that is unique across the entire PDS.

6.2 Collections

Products are grouped into *collections* by type and content. Observational data, for example, will be gathered into an observational data collection; documents into a document collection; and supplementary data into a supplementary collection. Most archives are large enough that there will likely be many observational data collections, and possibly several document and supplementary collections. Observational data collections are typically split along instrument, instrument mode or data type (image vs. spectra) boundaries. The products included in a single collection should, in general, be very closely related.

To define a collection, the data preparer creates a table file listing of all the PDS label files and their logical identifiers (see below) that are part of the collection, and then creates a collection label, which closely resembles a simple product label, to describe this table. So a collection can be thought of as a higher-order product – one that provides a link between products that share common characteristics. The collection label will also contain a logical identifier for the collection that is unique across the entire PDS.

The products listed in the collection table must be physically located either in the same directory as the collection label, or in one or more subdirectories of it.

6.3 Bundles

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

As with collections, bundles are defined by creating a table that lists the label for each constituent collection and the collection's logical identifier. Then a label resembling a simple product label is prepared to assign a logical identifier to the bundle and define the common characteristics of the collections comprising the bundle.

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

6.4 Logical Identifiers

Every product, collection and bundle in the archive will have its own unique *logical identifier* or *LID*. Here's what a LID might look like for an observational data product from the fictional "Sunburn" mission:

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

The `URN:NASA:PDS` part is required in all PDS LIDs. It identifies the LID as a PDS system reference. The next segment, `Sunburn-1`, is the ID for one of the bundles produced by the *Sunburn* mission – the one containing this data product. All the collections in this bundle will have the bundle ID as part of their LID. The next segment, `SolCam-Cr1-Raw`, is the ID of the particular collection the product belongs to. All products in this collection will have this collection ID as part of their LID. Finally, the `Image01321` part is an ID unique (within the `SolCam-Cr1-Raw` collection) to the product. By incorporating the bundle and collection IDs into the product IDs, data preparers can be sure of creating product LIDs that will be unique across the entire PDS, and in fact unique across the entire universe of LIDs from archive organizations.

Note that the LID or logical identifier is not tied to the physical location of the data – thus it is a *logical* identifier, as opposed to a file specification. We use *LID* or *logical identifier* in this document whenever we mean the full LID, from `URN:` to the last data preparer-specified segment. We use the term *ID* to refer to a segment (i.e., a string between colons) of a logical identifier associated with a particular level in the LID, like the segment associated with the bundle ID in the example above.

7. Overview of Data Development Strategies

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

7.1 Preliminary Steps

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

1. *Gain some familiarity with XML.* Your consulting PDS node can supply you with working label and schema examples to play with if you want to dive directly into the PDS end of the

pool, but any introductory tutorial on XML syntax will provide a good basis for PDS work with XML.

2. *Find an XML-aware editor.* There are shareware and commercial editors available for all major platforms. Your PDS consultant can suggest one or two to check out if you don't have a local XML guru to advise you. In particular, familiarize yourself with the validation capabilities of your editor.

3. *Gain some familiarity with PDS data.* Search the current holdings for similar data to what you plan to prepare, and see what sort of data structures and additional documentation are provided above the minimal PDS requirements.

7.2 Understand the Data

For each instrument producing observational data, consider what sequence of arrays and tables will best contain the data. Consider what additional information should also be included in the product labels to document the observations. For large missions, consider what attributes should be defined uniformly across instruments and make plans to agree on definitions and value formats early in the design process.

Also think about the supplementary products that may be provided with the observational data, including both visualization tools like browse directories as well as support files like observation logs, command sequences and look-up tables. Finally, consider what higher-order and derived products might be produced.

7.3 Design the Products

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

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

Only the most general characteristics appear in the PDS global classes, and those and the node-level classes will be provided in

the tailored schemas provided to the data preparer as starting points. So the major work of the data preparer in designing product labels comes from determining what additional information needs to be included in the various product labels, and defining the attributes and classes needed to hold them in the local data dictionary.

The output of this effort will be the schemas and/or label templates passed to the pipeline programmers for creating the PDS labels.

7.4 Design the Collections

Closely related products are gathered into collections. The major design process for collections is simply deciding which products are to be included in what collections. PDS requires that collections contain only one type of product – observational data, documents, or supplementary data – but after that the decisions are left to the data preparer. The vast majority of data preparers divide data into collections based on things like mission phase, target, instrument, data reduction level, and so on. So one collection may contain all the raw images recorded by the navigational camera during the second cruise phase, for example.

Broadly speaking, collections should be defined to group products in a way that makes sense from an end-user point of view. In other words, we would expect that it would often be the case that a user who wants one product from a collection will also want other products from the same collection, so it makes sense from an end-user perspective to group products by target, instrument, reduction level, etc. Your PDS node will be happy to provide specific advice for your situation if it is not clear what divisions might "make sense" to an end-user of your data.

7.5 Design the Bundles

Bundles, unlike collections, will generally be transparent to end-users. Collections are gathered into bundles mainly to facilitate data management and accounting. Consequently, the criteria for what collections should go into a bundle vary a bit more widely than those for gathering products into collections.

Popular criteria for designating bundles include:

- All collections from a particular time period or mission phase
- All collections related to a particular target
- All collections produced by a specific instrument
- All collections peer-reviewed at the same time

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

7.6 Final Notes

Missions and similar large data providers obligated to archive with PDS will generally create a document defining the archive design, designating responsible parties and laying out a schedule for reviews and deliveries. This document might be called an archive plan, an interface control document, a system interface specification, or something similar. It will generally require sign-off from both the data provider and the PDS. Your consulting PDS node can provide examples of existing documents for you to follow.

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

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

II. PDS4 Data Structures

8. Scalar Storage Formats

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

This section describes the scalar types that may be used for observational data (i.e., in arrays and tables).

8.1 Character Types

NOTE: DDWG still needs to discuss this material and finalize decisions on details such as: ASCII vs. UTF-8 in tables that use character data; prohibiting character pixels in arrays; delimiters in ASCII table files; and character sub-type names (for dates, times, etc.).

Character fields in observational data files are restricted to the printable 7-bit ASCII characters, plus the blank character. PDS defines a number of specific types based on formatting constraints and requirements. These types may be used in both binary and character tables; these types *must* be used exclusively in character tables.

PDS defines the following character types:

[The final CHARACTER TYPE LIST will be inserted here.]

Character table files will also have line delimiters, but the line delimiters are not considered part of any table field.

8.2 Binary Types

Binary types may be used as the element type of an array or a field type in a binary table file. The binary scalar types allowed in PDS4 data files are:

[The final BINARY TYPE LIST will be inserted here.]

9. Arrays

NOTE: This section will require heavy editing when we finalize the appearance of the array definition in the various image and other classes that might be based on it – or perhaps it would be better written from a lower-level perspective without reference to data objects at all (to introduce the concept of the four basic data structures, for example).

The simplest structure available for data storage is the homogeneous array, that is, an array in which all the elements are the same data type. This is the structure underlying all images, spectral cubes, maps and similar structures. To define an array you must first define the element type, and then define the number and lengths of each of the axes.

9.1 Elements

Any of the binary types listed in section 8.2 may be used as the element type for an array. In addition, elements may be associated with a number of characteristics like units of measure, an offset, or a scaling factor. The data preparer can also designate specific flag values to be used, for example, to indicate saturated pixels in an image. In the PDS4 XML label, the element and its attributes will be defined in an element class in the global PDS data dictionary.

9.2 Axes

Each array will have one or more axes. In the PDS4 label, each axis will be defined in its own class, so that interpretive attributes (like the name of the axis) can be specified for each axis individually. The principal attribute of an axis is its length. The order in which the axes are defined is significant; it is important to know which axis comes first, which second, etc., so that axes can be accurately mapped to subscripts in storage arrays in program memory, and to storage order in the physical file. In the PDS4 label, axis order will be stated explicitly via a sequence number in each axis class. If you think of the axes of an array corresponding to subscripts, then the axis with sequence number 1 would be the first (leftmost) subscript, axis number 2 the next, and so on. The axis class is also defined in the global PDS data dictionary.

9.3 Storage Order

The individual elements of any array are stored with their bytes in the order dictated by their scalar type.

The elements of the array are stored such that the subscript along axis 1 (notationally, the leftmost axis) varies most rapidly; that along axis 2 next most rapidly; and so on. This means that the elements along the highest-numbered (rightmost) axis will be stored contiguously. This storage order has also been called

column-major order historically, from 2-dimensional arrays in which the subscripts were thought of as [*row, column*] - though this becomes a less intuitive description as the number of axes increases.

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

9.4 Additional Considerations

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

10. Tables

NOTE: This section may need to be modified depending on the outcome of discussions regarding UTF-8 vs. ASCII in data files. It is currently written assuming only ASCII will be allowed in character tables. We also need to decide whether we want to allow columns which are vectors (i.e., corresponding to PDS3 ITEMS), and what the record delimiter will be. We also need to consider a special data type to use for spare bytes in binary files.

And, of course, if we prefer to address the four basic storage structures in this section, then this needs to be rewritten with that in mind.

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

Tables come in two flavors: binary and character. Binary tables may contain character fields, but character tables may never contain binary fields.

10.1 Character Tables

Each field in a character table must be of one of the character data types listed in section 8.1 Character Types. Binary types are strictly forbidden. Each record in a character table must end with a two-byte record delimiter consisting of the carriage return character followed by the linefeed character.

Character tables will usually have *gutter space* between columns. In most cases this will consist of a blank character, but it can be a string of blanks, or any other string of one or more valid characters (printing characters plus the blank). Gutter space is not required, although character tables without gutter space tend to be less readable.

Gutter space, where it exists, and the two-byte record delimiter are considered to be outside the actual fields; their bytes should

never be included in any of the field definitions in the character table record. So if you add up the widths of all the defined fields, the difference between that sum and the total record length should be the number of bytes occupied by the gutter space and the record delimiters.

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

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

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

Because numeric values expressed as ASCII strings are not constrained by the limitations of binary conversion or byte counts, character tables can potentially contain integer values greater than those in binary integers, and real numbers with more decimal places of precision than possible in binary floating point formats. Of course, the creation and manipulation of such values still requires a binary conversion, but if an observer creates hyper-precision values and wants them preserved, character tables may be the only option.

Despite the fact that they generally require more bytes to store the same information as a binary file, character files are often preferred by users because of human-readability, the prospect of being able to read the files directly into spreadsheet programs, and the ability to read character data into every major programming language without having to worry about detailed architectural issues like byte order on the target machine.

10.2 Binary Tables

Unlike character tables, binary tables must have no record delimiters and no gutter space. All the bytes in a binary record must be accounted for in the field definitions. If a record contains *spare* bytes (bytes not currently filled with data), these should be defined as a column with a data type of "SPARE".

The fields in a binary file will mainly be of the binary types listed in section 8.2, although some of the character field types listed in 8.1 are fairly common – dates and times, for example, as well as simple string fields to contain flags or time tags.

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

1. For tables containing primarily numeric data, the binary version will likely be substantially smaller than the character equivalent; and
2. There is no conversion error in reading binary floating point numbers into memory on machines that use the same floating point storage format as PDS4.

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

The IEEE-954 floating point format required by PDS4 for real numbers is fairly common across available platforms as of this writing, so for the average user the major considerations for reading binary data are byte-ordering and byte-alignment.

10.3 Binary vs. Character Tables

In general, users tend to prefer character tables for most data because of the high degree of confidence they have in being able to read the data into their analysis programs properly. From an archiving point of view, character tables are extremely stable, so most nodes prefer character tables as well.

When the data consist entirely of large quantities of numbers – to the point where record and file size become ungainly for visual inspection and common spreadsheet applications – binary tables present a good alternative.

When deciding whether to use binary or character tables for a PDS4 data product, remember that your PDS consulting node is thinking about long-term preservation and end-user preferences, not relative efficiency in any one processing environment. When binary tables are the right answer, you should include a character equivalent of a few records from a representative file that can be used to validate I/O in programs trying to read the binary table data product.

10.4 Defining Fields

NOTE: This needs to be expanded depending on how the final field definition system shakes out.

In the PDS4 labels, the table itself is a class and each field in the record is a subclass within the table class. Each field has a name, a data type and a start position within the record. Additional optional attributes include things like units of measure, offset and scaling factors, descriptions, and constants used to indicate missing data.

11. Documents

NOTE: We need to decide a couple things before this section can be reasonably completed. First, we need to finalize a general document labeling scheme. Second, we need to decide what we want to do, at least for the first public release, about HTML documents. Actual HTML is problematic as an archival format, and the conversion to XHTML is

non-trivial. We also need to think through the implications of accepting anything other than PDF/A and UTF-8 for "archiving".

The word *document* is used here to describe "traditional documents" – that is, straight text that might be read either online or by printing out a hard copy. Non-traditional documents fall into the *supplementary information* category, described in the following chapter.

Documents must be submitted for archiving in one of the following forms:

1. PDF/A
2. Flat (i.e., no graphics or special formatting), UTF-8 text files

11.1 Required Documents

PDS requires a minimum set of documentation with all data to provide details on how the data were obtained, calibrated or otherwise manipulated prior to archiving. Documentation considered essential to understanding or using the data must be submitted as part of the archive.

What constitutes the "minimum set of documentation" will vary from archive to archive. At the very least, all data preparers will have to prepare descriptions of the components of the observing system. These descriptions can be simple UTF-8 text files written specifically for the archive, or they might be republications of, for example, instrument description papers published in the refereed literature.

The data preparer and consulting PDS node should agree on a list of required document early in the design process, so that that list can be included in the archive plan.

Finally, the PDS will also accept additional documents - beyond those required for the archive – provided they are in one of the required formats.

11.2 PDF/A

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

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

11.3 Flat UTF-8 Text

UTF-8 is a specific implementation of the Unicode character standard.

The Unicode standard assigns a unique number, called a *code point*, to every character, glyph and control code used in text files. A Unicode Translation Format (UTF) maps Unicode code points to a specific byte value sequence. There are several UTFs

in use, including the most popular variants UTF-8, UTF-16 and UTF-32. PDS has designated UTF-8 as the text standard for PDS4. The move to Unicode text will allow the use of letters with diacritical marks as well as many single-character symbols (degrees, the Angstrom symbol, Greek letters, etc.) useful in text documents describing the archives. UTF-8 has several additional features that make it an appealing choice for PDS4 text files:

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

A *flat* text file is one that includes no graphics or images, references no external files, and contains no formatting beyond indenting and line breaks

11.4 Additional Formats

NOTE: We need to formalize this. Are we really going to archive non-PDF/A source files? What does that mean for migration in future?

Documents archived in PDF/A format will be created in some other format first and then converted. Data preparers may also submit the original source file for inclusion in the archive. The PDF/A version will be considered the master copy for archive purposes, but the alternate version can be made available to contemporary users on request for as long as that format is generally supported.

11.5 Labeling Documents

[This section is TBD until the labeling scheme is closer to beinf finalized..]

12. Supplementary Data

Supplementary Data comprises any information supplied with the observational data that is not a traditional document in the sense of the preceding section. It may include things like observing log tables, calibration images, filter profiles, sample spectral plots, and so on.

12.1 Required Supplementary Data

For any archive, some supplementary information may be required. For example, if the data are being supplied as binary

tables, PDS will generally require a sample ASCII translation of a few records from one file for use in validating I/O routines. Supplementary data requirements will be worked out between the data provider and the consulting PDS node during the archive design process.

12.2 Labeling Supplementary Data

Supplementary data files should be labeled in a manner appropriate to the data. If the supplementary data is a filter response table, for example, it should be formatted as a fixed-width character table and labeled accordingly. If it is a sample plot, it should be converted to PDF/A and labeled as a document.

[Here would be an excellent place to comment on supplementary info that comes in the form of an HTML web site, if we decide to accept that format.]

Following is a brief outline of proposed additional sections.