JPL D-7669, Part 2

Planetary Data System Standards Reference

Version 1.18.0

March 31, 2022

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Contents

Contents 2
1 Introduction 7
1.1 Purpose 7
1.2 Scope 7
1.3 Audience 7
1.4 Document Organization 8
1.5 External Standards 8
1.6 Document Availability 9
2 Archive Organization 10
2A Archive Logical Organization 10
2A.1 Bundles 10
2A.2 Collections 10
2A.3 Products 11
2A.4 Primary and Secondary Members 11
2A.5 Collection Types 11
2B Archive Physical Organization 13
2B.1 PDS Data Storage and Access 13
2B.1.1 Objects and Files 13
2B.2 Data Transfer 13
2B.2.1 Transfer Media 13
2B.2.2 Directory Structure 13
3 Labels 18
4 Fundamental Data Structures 20
4A Array 20
4A.1 Storage Order and Index Order - Definitions 20
4A.2 Storage and Index Order - Conventions in Popular Software Environments 21
4A.3 Array Storage Elements 21
4A.4 Axis Meaning 22
4A.5 Display Orientation 22
4B Table Base 23
4B.1 Fields 23

4B.1.1 Field Length ... 23
4B.1.2 Field Formats ... 24
4B.2 Groups ... 26
4C Parsable Byte Stream ... 26
4C.1 Delimiter Separated Value Format Description ... 27
4C.2 Delimited Tables ... 28
4D Encoded Byte Stream ... 29
5 Data Types ... 30
5A Attribute Value Types ... 30
5A.1 Boolean Types ... 30
5A.2 Date and Time Types ... 30
5A.3 Numeric Types ... 33
5A.4 String Types ... 35
5B Character Data Types ... 38
5C Binary Data Types ... 39
5C.1 Integers ... 39
5C.1.1 Signed LSB Integers ... 39
5C.1.2 Unsigned LSB Integers ... 41
5C.1.3 Signed MSB Integers ... 43
5C.1.4 Unsigned MSB Integers ... 45
5C.2 Reals ... 47
5C.3 Complex ... 50
5C.4 Bit Strings ... 52
5C.4.1 Unsigned Integers Stored as Bit Strings ... 52
5C.4.2 Signed Integers Stored as Bit Strings ... 53
6 Naming ... 54
6A Character Sets ... 54
6A.1 ASCII Character Set (also known as IsBasicLatin) ... 54
6A.2 ASCII Alphanumeric Character Set ... 56
6A.3 ASCII Printable Character Set ... 57
6A.4 UTF-8 ... 57
6B Namespace ... 57
6B.1 Namespace Creation and Use ... 57

6B.2 Formation of Namespace IDs: 58
6B.3 Formation of Namespace URIs: 58
6C File and Directory Naming 59
6C.1 File Names 60
6C.1.1 Rules 60
6C.1.2 Prohibited File Names 60
6C.1.3 Reserved File Names 60
6C.1.4 Prohibited Base Names 61
6C.1.5 Reserved Base Name Components 61
6C.1.6 Reserved File Name Extensions 61
6C.2 Directories 63
6C.2.1 Rules 63
6C.2.2 Reserved Directory Names 63
6C.2.3 Prohibited Directory Names 63
6C.2.4 Directory Path Names 64
6D Identifiers 64
6D.1 Local Identifier 64
6D.2 Logical Identifier 64
6D.3 Versioning 65
6D.3.1 Bundle, Collection, and Product Versioning 65
6D.3.2 System, Namespace, Data Dictionary, and Document Versioning 66
6E Classes, Attributes, and Attribute Values 67
6E.1 Classes 67
6E.2 Attributes 67
6E.3 Attribute Values 68
6E.3.1 Enumerated Attribute Values 68
6E.3.2 Attribute Value Units 68
6E.3.3 Attribute Value Limits 68
7 Units 70
7A Types of Measurement 70
7B Specific Units 71
7C Prefixes 71
7D Derived Units 71

7E Expressions ... 72
8 Content ... 73
8A Documentation Requirements ... 73
8A.1 Labels ... 73
8A.2 Internal Documentation ... 73
8A.2.1 Document Collections ... 74
8A.2.2 Document Products ... 74
8A.3 External Documentation ... 75
8B Context Bundle, Collections, and Products ... 76
8B.1 Collections Under the PDS4 Context Bundle ... 76
8B.2 PDS Context Products ... 76
8B.3 Context Collections and Products in Science Bundles ... 77
8C Calibration ... 77
8D Geometry ... 77
9 Special Constraints ... 78
9A Header Class ... 78
9B Group Fields and Groups ... 78
9C Product_Collection ... 78
9C.1 Creating an Inventory Table ... 78
9C.2 Generating and Populating a Product_Collection Label ... 79
9C.3 Constraints on Collections ... 80
9C.3.1 Mission Science Data Collection - Constraints ... 80
9D Product_Bundle ... 80
9D.1 Creating a Product_Bundle ... 80
9D.2 Generating and Populating a Product_Bundle Label ... 80
9D.3 Constraints on Bundles ... 81
9D.3.1 Mission Science Archive Bundle - Constraints ... 81
9E Product_Native ... 81
9F Class Selection ... 82
9G Product_Metadata_Supplemental ... 83
9H Composite Structure ... 84
10 General Policies ... 84
Change Log ... 85

Version 1.0.0 85
Version 1.2.0 85
Version 1.3.0 90
Version 1.4.0 90
Version 1.7.0 93
Version 1.8.0 97
Version 1.9.0 97
Version 1.10.0 98
Version 1.10.1 98
Version 1.11.0 98
Version 1.12.0 99
Version 1.13.0 99
Version 1.14.0 100
Version 1.15.0 100
Version 1.16.0 100
Version 1.17.0 101
Version 1.18.0 101

1 Introduction

Version 4 of the Planetary Data System (PDS) is now available. Data from new missions are being designed for ingestion into PDS4' and earlier PDS3 data sets are being migrated to the new PDS4 system[1].

The PDS4 Information Model (IM) is the fundamental reference for PDS4 structure; its requirements can be validated automatically using eXtensible Markup Language (XML) schemas.

The PDS4 Data Dictionary Data Base (DDDB) is the fundamental reference for definitions of classes and attributes.

The PDS4 Concepts document provides an overview of the philosophy behind and organization of PDS4. It includes a glossary of terms as used within PDS4.

1.1 Purpose

The PDS4 *Standards Reference* (*SR*) is a compilation of policies, rules, and other PDS4 constraints that are not given explicitly in the IM and DDDB. The combination of the IM, DDDB, and SR gives the set of common requirements that apply across PDS. Each PDS discipline node (DN) may establish additional requirements that further constrain — but do not conflict with — the common requirements and apply only to data ingested through that node.

1.2 Scope

The PDS4 *SR* applies to PDS Version 4 and its holdings. External standards, adopted by PDS, are included by reference. Examples and recommended best practices can be found in the PDS4 *Data Provider's Handbook* (*DPH*) and references therein.

1.3 Audience

The PDS4 *SR* is intended primarily to serve the community of scientists and engineers responsible for preparing planetary science data for submission to PDS4. Such submissions include restored data from the era prior to PDS or from earlier versions of PDS, mission data from active and future planetary missions, and data from Earth-based sites, including laboratories and independent research efforts. The audience includes personnel at PDS discipline and data nodes, principal investigators and their staffs, and ground data system engineers. The document will be most useful to people who have experience with PDS archiving; those new to PDS may find the *PDS4 Concepts* document to be a better place to start.

While instructions and examples in this document are given in the context of data prepared for archiving in the NASA Planetary Data System, the PDS4 Standards are available for use by other

[1] PDS4 standards are not backwardly compatible with version 3 (PDS3). In principle, version 4 data can be described using version 3 labels, but the converse is not true; some PDS3 structures are no longer supported under PDS4.

national and international space agencies, and indeed the PDS welcomes such usage. Other agencies using the PDS4 Standards have complete governance over their own data repositories, name spaces, and mission and discipline data dictionaries. Readers are encouraged to contact their archiving authority for details.

1.4 Document Organization

The PDS4 *SR* is divided into parts. The first, Sections 1-7, contains detailed information on the structural aspects of a PDS4 archive: the format of data, labels, and their assembly into 'packages' for submission and transfer. The second, Section 8, is more focused on the nature of *what* is stored in PDS4 archives — such as adequacy of calibration and documentation. The third, Section 9, documents additional constraints which do not fit easily into the previous two sections. The fourth, Section 10, provides a link to PDS policy statements with which this and other documents must comply.

1.5 External Standards

External standards, which apply to this document and to PDS4-compliant data, include the following:

American National Standards Institute (ANSI)

- ANSI INCITS 4-1986 (R2007) *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

Astrophysics Data System

- https://adsabs.github.io/help/actions/bibcode Information on the ADS BibCode Format

Consultative Committee for Space Data Systems (CCSDS)

- CCSDS 641.0-B-2 *Parameter Value Language Specification (CCSD0006 and CCSD0008)* (also available as ISO 14961:2002)

Institute of Electrical and Electronics Engineers (IEEE)

- IEEE 754-2008 *Standard for Binary Floating-Point Arithmetic*

International Standards Organization (ISO)

- ISO 646:1991 ISO *7-bit coded character set for information interchange*
- ISO 8601:2004 *Data Element and Interchange Formats – Representations of Dates and Times*
- International Standards Organization/International Electrotechnical Commission (ISO/IEC) 10646:2012 *Information technology – Universal Coded Character Set (UCS)*
- ISO/IEC 11179-3:2003 *Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes*
- ISO/IEC 11404:2007 *General-Purpose Datatypes (GPD)*
- ISO/IEC 19757-3:2006 *Information technology -- Document Schema Definition Languages (DSDL) -- Part 3: Rule-based validation -- Schematron*
- ISO 14721:2003 *Open archival information system – Reference model*
- International Standards Organization / Technical Standard (ISO/TS) 15000-3:2004 *Electronic business eXtensible Markup Language (ebXML) – Part 3: Registry*

information model specification (ebRIM)

- ISO/TS 15000-4:2004 *ebXML – Part 4: Registry services specification (ebRS)*

National Institute of Standards and Technology (NIST)

- NIST Special Publication 330 *The International System of Units (SI)*, United States version of the English text of the eighth edition (2006), Issued March 2008

World Wide Web Consortium (W3C)

- *Extensible Markup Language (XML) 1.0* (Fifth Edition)
- *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures* (W3C, 2012a)
- *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* (W3C, 2012b)

Requests for Comment

- RFC 1321, The MD5 Message-Digest Algorithm (http://tools.ietf.org/html/rfc1321)
- Yergeau, F., *UTF-8, A Transformation Format of ISO 10646*, RFC 3629, Internet Engineering Task Force, November 2003 (http://tools.ietf.org/html/rfc3629)

International DOI Federation

- http://www.doi.org General information and "Resolve a DOI Name" interface

1.6 Document Availability

PDS4 documents governing archive preparation are available online at:

https://pds.nasa.gov/datastandards/documents/

For questions concerning these documents, contact any PDS data engineer or contact the PDS Operator at pds_operator@jpl.nasa.gov or 818-393-7165.

Associated schemas for current and past versions of PDS4 can be found at

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

2 Archive Organization

2A Archive Logical Organization

2A.1 Bundles

A *bundle* is the default logical construct for archiving digital data in the PDS. (Recall that terms such as bundle, collection, and basic product are defined in the glossary of the *PDS4 Concepts* document.)

Bundles have a simple hierarchical structure. A bundle has one or more member *collections*, each of which has one or more member *basic products* (Figure 2A-1). PDS does not impose requirements on how bundles are defined except that (1) each bundle must have a unique identifier within the overall holdings of PDS, and (2) each bundle must be approved by a PDS peer review.

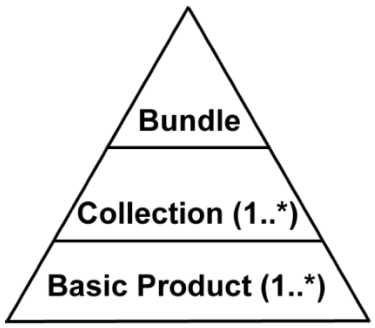


Figure 2A-1: Archive structure.

Members of a bundle are listed in a Product_Bundle, an XML file which serves as both a label and the bundle inventory. Product_Bundle is described and uniquely identified using the Product_Bundle class definition (see Section 9D for information on constructing Product_Bundle).

An optional "readme" file may be included as part of a Product_Bundle; it is described by the bundle label so is not a separate product. The "readme" file provides a general overview of the bundle contents and organization in human readable format. It may also contain general instructions for use of the bundle and contact information for the data provider or discipline node personnel. The "readme" file must be formatted either as 7-bit ASCII text or as UTF-8 text.

2A.2 Collections

Basic products are organized into collections based on the type and function of the data. PDS imposes only broad requirements on how these type and function boundaries are drawn (see Section 2A.5). Collections must be distinct within a bundle, products must be distinct within each collection, and each collection must be approved by a PDS peer review. Any single version of a collection is defined and uniquely identified using the Product_Collection class definition —

an XML label file paired with an inventory table, which lists collection members for that version (see Section 9C for information on constructing Product_Collection).

Collection members must use either a ".xml" or ".lblx" extension for the XML label files; but not a mix within a single Collection.

2A.3 Products

A *basic product* is the simplest product in PDS4 — one or more data objects and their description objects, which constitute a single observation, document, etc. Typically, a *data object* is a file containing a single image, table, or time series; *description objects* are typically text that describes both the format and content of the associated data object. A *label* is an XML file, which is the concatenation of one or more closely related *description objects* (such as for the red, green, and blue components of a color image) with some XML overhead; the corresponding basic product is that label plus the red, green, and blue image data objects. A document basic product is constructed in the same way: the basic product is the set of files containing text, figures, and tables together with a label comprising the description objects. Certain XML files qualify as basic products by themselves — they are 'XML documents'.

Digital objects which comprise observational data may be used in one and only one product.

Product_Collection and Product_Bundle are *aggregate products*; they define an aggregation of basic products and an aggregation of collections, respectively. They are not basic products. All products, whether basic or aggregate, must have globally unique logical identifiers (see Section 6D).

2A.4 Primary and Secondary Members

Basic products may be either *primary* or *secondary* members of their respective collections. A primary member is one that is being registered with PDS for the first time. A secondary member is one which is already registered with PDS, but which is now associated with an additional collection[2]. A product's member status (primary or secondary) is based on its *first* association with a collection. Although the product may be omitted from a later version of the collection, it retains its primary or secondary member status through all subsequent versions of the collection based on its initial association. In a similar way, collections are categorized as having either primary or secondary 'member status' in their bundles.

2A.5 Collection Types

Basic products must be organized into collections based on the type and function of the data. Data providers are expected to assign products to appropriate collections (*e.g.*, only 'quick-look'

[2] Data providers are not obligated to include physical copies of secondary member products or collections when they deliver bundles to PDS, since the secondary members already exist elsewhere within PDS. PDS, on the other hand, must deliver copies of these secondary members when bundles are distributed (unless waived by the recipient).

or ‘browse’ products are expected to be assigned to a *browse* collection); the assignments will be considered during peer review. PDS recognizes the following types of collections.

A *browse* collection contains ‘quick-look’ products designed to facilitate use of the archive. Products in a browse collection are defined using appropriate classes; possibilities include Product_Browse and Product_Thumbnail.

A *calibration* collection contains data and files necessary for the calibration of basic products. Products in a calibration collection are defined using the appropriate class; possibilities include Product_Observational and Product_Document.

A *context* collection is the list of products comprising various objects, identified within the PDS4 registry, that are specific to the science bundle. These include physical objects such as instruments, spacecraft, and planets and conceptual objects such as missions and PDS nodes. All products in a context collection will be secondary members of the collection, since the primary versions are curated by the PDS Engineering Node or other responsible agency.

A *data* collection contains observational products, often separated according to processing level, target, instrument mode, etc.; these are the science data that most users seek. Products in a data collection will typically be defined using the Product_Observational class.

A *document* collection includes all components (figures, tables, text, etc.) of one or more related documents. The document collections included with a bundle are those deemed useful by the data preparer and consulting node for understanding, interpreting, and using other collections in the bundle. Documents may include software interface specifications (SISs), calibration reports, and data acquisition summaries, among other topics. Products in a document collection will be defined using the Product_Document class.

A *geometry* collection contains non-SPICE geometry products — for example, Supplementary Experiment Data Record (SEDR) data, gazetteers, tables of anaglyph pairs, or footprint files. Detailed information about particular cartographic projections utilized in the archive may also be included. Products in a geometry collection are defined using an appropriate class; possibilities include Product_Observational, Product_Ancillary, and Product_Document.

A *miscellaneous* collection contains supplementary information deemed by the data provider to be useful in the interpretation and use of other collections in the bundle, but which does not fit within the scope of the other collections. For example, meta-data catalogs, data base dumps, and records of modification history could be included. Because of the disparate nature of the files included in a miscellaneous collection, no single class exists for miscellaneous products. The following are some of the possibilities: Product_Ancillary, Product_Document, Product_File_Text, Product_Observational, Product_Thumbnail, and Product_Zipped.

An *XML schema* collection contains all XML schema files included in or referenced by XML labels in the bundle along with any Schematron files created for validation purposes. A bundle is required to contain an XML schema collection unless all of the collection members would be secondary in which case the XML schema collection becomes optional. There can be no more than one XML schema collection per bundle.[3]

---

[3] The PDS4 schema and Schematron files are software-readable representations of the PDS4 Information Model. They are discussed in section 3, Labels.

A *SPICE kernels* collection contains individual SPICE files and their XML labels, organized by kernel type. Products in a SPICE kernel collection must be defined using the Product_SPICE_Kernel class.

2B Archive Physical Organization

2B.1 PDS Data Storage and Access

Except for file and directory naming rules in Section 6C and the restrictions on objects and files in Section 2B.1.1 immediately below, no standards are set within this document for the organization of the physical archive (*i.e.*, there are no requirements for how the data are stored within or accessed from a physical repository or data storage).

2B.1.1 Objects and Files

PDS requires that each digital object be physically distinct and contiguous in a single file; objects may not overlap. For example, a color image may be stored as three separate Array_2D_Image (RGB) objects with each of the R, G, and B components being stored separately. The lines or samples of the three components may not be interleaved in this case. However, it is permissible to define such an image as a single Array_3D_Image object with interleaved lines or samples.

Although a single file may contain multiple digital objects, no digital object may extend beyond a single file. For example, the R, G, and B image components above may each be stored in a separate file, or they may all be stored together in a single file, but any one component may not span two or more files.

A label and all of its associated digital objects must be stored in the same directory unless the directory_path_name attribute is available, in which case directory_path_name may be used to indicate a subdirectory relative to the directory containing the label.

2B.2 Data Transfer

The parties involved in a data transfer to, from, or within PDS are free to adopt any procedure that is mutually acceptable. In lieu of such an *ad hoc* agreement, the following procedure is suggested.

2B.2.1 Transfer Media

The parties involved in a data transfer must agree on the physical media and interfaces to be used (CD, DVD, magnetic disk, electronic, etc.). PDS defers to industry standards for the media selected.

2B.2.2 Directory Structure

The directory structure described in this section is given as an example of how data may be packaged for transfer. PDS4 bundles and collections are not required to follow this structure.

Using this method, data may be transferred using a directory structure that *parallels* the logical organization of the bundle (Section 2A). Note that the directory structure does not itself have to

meet the requirements for logical bundles or collections. Many transfers will contain only a portion of a bundle or collection.

The following sections describe such a directory structure, starting with the root directory.

In the tables and figures below, the following conventions have been adopted:

- **Bold-faced** font is used for directory names, while unbolded font is used for file names. Where no pattern for directory or file name construction is specified, example names are shown in *italics*.
- The asterisk "*" indicates that any text may be substituted. In Table 2B-1 **browse[_*]** means that "**browse**" or any directory name starting with "**browse_**" is acceptable. See Section 6C for restrictions on directory and file names.
- Square brackets "[ ]" identify variable parts of file or directory names. For example, readme[_*].txt in Table 2B-1 means that the file name beginning with "readme" may be lengthened with any character string that begins with an underscore "_". Using an underscore for readability is a common practice, but a hyphen or any other legal file name character could be used.

2B.2.2.1 Root Directory

The root directory corresponds to the top level of a bundle. It contains a Product_Bundle and subdirectories corresponding to its member collections.

Table 2B-1 – *root* Directory

File or Directory Name	**Notes**
root	may be unnamed
bundle[_*].xml | .lblx	See restrictions
readme[_*].txt	described by the bundle[_*] XML label
browse[_*]	
calibration[_*]	
context[_*]	
data[_*]	
document[_*]	
geometry[_*]	
miscellaneous[_*]	
xml_schema[_*]	
spice_kernels[_*]	

A bundle is described by an XML label with the name "bundle[_*] and having an extension of either ".xml" or ".lblx". One or more of these files may be included under the root of the transfer directory structure, but none is required.

The optional "readme" file provides a general overview of the bundle contents and organization in human readable format. The file is an optional component of Product_Bundle and is described in the bundle label.

2B.2.2.2 Subdirectories

Beneath the ***root*** directory in this example structure, the top-level subdirectories have a one-to-one correspondence with the bundle's collections. Each collection subdirectory has a collection label with a name of the form "collection[_*] and having an extension of either ".xml" or ".lblx". and a collection inventory table with a name of the form "collection[_*].csv". (Sometimes "collection[_*]_inventory.csv" is used.) See Section 9C.1 for details on construction of collection inventory tables.

Collection labels and inventory tables usually reside in the top-level subdirectories. If there are many products in the collection they are usually divided into lower level subdirectories as shown in Figure 2B-1. The lower level subdirectories are named and organized at the discretion of the data provider except for SPICE kernel subdirectories, which have a required structure (see Section 2B.2.2.3). An example directory structure may be found on the PDS4 Examples web page at https://pds.nasa.gov/datastandards/documents/examples/.

In cases where a bundle contains multiple collections of the same type, the names of the subdirectories containing these collections are distinguished by a suffix to the subdirectory name. For example, if two calibration collections are present in the archive, the two calibration subdirectories might be named **calibration_*flight*** and **calibration_*ground*** (Figure 2B-2).

```
root
|       - bundle.xml
|       - readme.txt
|
|       - browse
|               - collection_browse.xml
|               - collection_browse.csv
|
|               - flight
|                       - flight_0001_1.0.jpg
|                       - flight_0001_1.0.xml
|                       - flight_0002_1.0.jpg
|                       - flight_0002_1.0.xml
|                               ...
```

Figure 2B-1: Illustration of an example structure for one **browse** subdirectory (one collection) with many basic products organized into subdirectories.

```
root
|       ...
|       ...
|       ...
|
|       - calibration_flight
|               - collection_calibration_flight.xml
|               - collection_calibration_flight.csv
|
|               - cruise
|                       - flightcal_product01.jpg
|                       - flightcal_product01.xml
|                       - flightcal_product02.jpg
|                       - flightcal_product02.xml
|                               ...
|
|               - orbit_0001-0100
|
|       - calibration_ground
|               - collection_calibration_ground.xml
|               - collection_calibration_ground.csv
|
|               - 2010_03
|                       -2010_03_04_0001.jpg
|                       -2010_03_04_0001.jpg
|                       -2010_03_04_0001.jpg
|                       -2010_03_04_0001.jpg
|                               ...
```

Figure 2B-2: Illustration of an example structure for two subdirectories containing two collections of the same type (calibration). Lower level subdirectories (***cruise***, ***orbit_0001-0100***, ***2010_03***, and possibly others) are defined and named by the data provider.

2B.2.2.3 SPICE Kernels Subdirectories

Under a SPICE Kernels subdirectory, normally named "spice_kernels", individual SPICE files and their XML labels must be organized by kernel type and placed in the corresponding subdirectories:

ck	subdirectory with CK files (spacecraft and instrument orientation data)
dbk	subdirectory with DBK files (data bases in SPICE format)
dsk	subdirectory with DSK files (digital shape data for natural bodies)
ek	subdirectory with EK files (events information)

fk	subdirectory with FK files (reference frames definitions)
ik	subdirectory with IK files (instrument parameters and FOV definitions)
lsk	subdirectory with LSK files (leapsecond information)
mk	subdirectory with MK files (meta-kernels listing kernels to be used together)
pck	subdirectory with PCK files (natural body rotation and size/shape constants)
sclk	subdirectory with SCLK files (spacecraft clock correlation data)
spk	subdirectory with SPK files (trajectory and ephemeris data)

Note that SPICE kernel files have specific naming requirements. In particular, there are mandated file name extensions based on the type of kernel file (see Section 6C).

3 Labels

PDS product labels are required for describing the contents and format of each individual product within an archive. Specifically, the labels contain the description objects that describe corresponding data objects. The description objects are populated using a standard set of classes, attributes, and standard values, which are themselves defined in the PDS4 Information Model. The PDS4 Information Model is expressed, and therefore PDS labels are written, in the eXtensible Markup Language (XML). Thus, a PDS label is an "XML document".

In order for any XML document (including a PDS label) to meet the XML standard, it must be both "well formed" and "valid". A well-formed XML document must have correct XML syntax; a valid XML document must conform to the rules of XML schema document(s) (XSD) and Schematron document(s) (SCH). XSDs are used to define the large-scale requirements for the labels — classes, attributes, and sequencing. Schematron is used for the fine control within lower level contexts — values and relationships between attributes within a single class.

Thus, in order for a label to comply with PDS4 standards, it must:

- have correct XML syntax
- be compliant with the class and attribute structures defined by the PDS and relevant discipline nodes and missions in their respective XSDs
- be compliant with the rules governing specific attributes and their values as set by the PDS and relevant discipline nodes and missions in their respective Schematron files
- be compliant with the rules governing both URI namespace constructs and fully resolvable schema location constructs to the PDS4 "released" repository for referenced XSD and SCH documents.
- have a file name ending with either the extension "xml" or "lblx".

PDS4 schemas are supplied to data providers by the PDS. Missions and other data providers must not modify these pre-existing schemas; however, they may extend existing classes and provide additional attributes in their own dictionary schemas, with the approval of a PDS discipline node. Schemas controlled by international agencies may be modified at the discretion of the stewards of those schemas, without the need for PDS approval.

Under PDS4, all product labels are detached from the files containing the digital objects they describe. There is one PDS4 label for every product. Each product may contain one or more data objects. The digital objects of a given product may all reside in a single file[4], or they may be stored in multiple, separate files. However, a single digital object may not reside in multiple files.

Figure 3-1 shows the general structure of a label, simplified to typical components. "XML Declaration and Schema Reference" is a few lines of XML overhead required for label implementation; the remainder of the label is defined by the PDS4 Information Model. `Root Tag`, `File_Area_Definition`, and `End Tag` are placeholders, which vary among

[4] Except for documents, in which case each object must be in a separate file.

labels; the first points to the data object, the second describes format and content of the associated physical file(s) if present, while the third marks the end of the label. `Identification_Area` provides 'fingerprints' and historical information about the product, `Observation_Area` provides information on how the data (or equivalent) were acquired. `Reference_List` points to other sources of information about the product that a user may wish to pursue. `File_Area_Definition` describes format and objects contained in the associated physical file(s).

```
XML Declaration and Schema Reference
Root Tag
        Identification_Area
                Alias_List
                Citation_Information
                Modification_History
        Observation_Area
                Time_Coordinates
                Primary_Result_Summary
                Investigation_Area
                Observing_System
                Target_Identification
                Mission_Area
                Discipline_Area
        Reference_List
                Internal_Reference
                External_Reference
                Source_Product_Internal
                Source_Product_External
        File_Area_Definition
                File
                Data_Object_Definition
End Tag
```

Fig. 3-1. Simplified label structure

Data providers should consult the Information Model for details and the *Data Provider's Handbook* for suggestions on how to construct actual labels. Most of the top-level components have nested classes that are not shown in Figure 3-1.

4 Fundamental Data Structures

There are four fundamental data structures that may be used for archiving data in the PDS. All products delivered to the PDS must be constructed from one or more of these structures. These four fundamental structures are described using four base classes: Array (used for homogeneous N-dimensional arrays of scalars), Table_Base (used for repeating records of heterogeneous scalars), Parsable_Byte_Stream (a stream of bytes that can be parsed using standardized rules), and Encoded_Byte_Stream (an encoded stream of bytes). All other digital object classes in the PDS are derived from one of these four base classes.

4A Array

The first of the four basic PDS4 structures is the Array. Any data structure that consists of fixed-length rows of homogeneous elements in any number of dimensions must be described using the Array class or one of its subclasses. As two- and three-dimensional (2-D and 3-D, respectively) arrays are among the most commonly used data structures in science, the discussion below will focus on them.

4A.1 Storage Order and Index Order - Definitions

The location of a particular element in a 2-D array is specified using its row number (first index) and column number (second index) when using standard matrix notation. Thus, in the array shown below, the position of the "2" (row 1, column 2) is specified using the notation (1,2), while the position of the "4" (row 2, column 1) is specified as (2,1).

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

An N-dimensional array is always stored in computer memory or in a data file as a linear sequence of numbers. There are two ways to store 2-D array data in linear memory: row-major order and column-major order. In row-major storage, the rows of the array are stored sequentially. In column-major storage, the columns are stored sequentially.

Thus, using the above notation, a row-major 2x3 array would be stored this way in memory:

(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)

or, using the numbers in the example above, the values would be stored as follows:

1 2 3 4 5 6

The same array, in column-major order, would be stored this way:

(1,1), (2,1), (1,2), (2,2), (1,3), (2,3)

or:

1 4 2 5 3 6

PDS uses the terminology *Last_Index_Fastest* as a synonym for 2-D row-major order and *First_Index_Fastest* as a synonym for 2-D column-major order. The PDS terms generalize immediately to N-dimensional arrays, whereas the meanings of "row" and "column" become ambiguous. In an N-dimensional array with Last_Index_Fastest, the next to last index varies next to fastest and the first index varies slowest.

Last_Index_Fastest (row-major) is the required storage order for PDS4 array data. Note that this does not require modifying data already stored in First_Index_Fastest (column-major) order; only the order in which the axes are defined in the label needs to be switched. See the discussion below under Axis Meaning for more information. For example, the PDS4 label for a two-dimensional array stored in First_Index_Fastest order should define the row axis first and the column axis second. The PDS4 label for an array stored in Last_Index_Fastest order should define the column axis first and the row axis second.

4A.2 Storage and Index Order - Conventions in Popular Software Environments

FORTRAN, IDL, and MATLAB arrays are stored in First_Index_Fastest (column-major) order; all other major programming languages store arrays in Last_Index_Fastest (row-major) order.

Both storage orders were permitted in PDS3 data, but Last_Index_Fastest (row-major) was the default and recommended storage order.

FITS format uses First_Index_Fastest storage order if one assumes that the shape of the array is defined by (NAXIS1, NAXIS2, ...). However, the popular FITS programming libraries cfitsio and pylab describe the arrays as Last_Index_Fastest, *i.e.*, with the shape (..., NAXIS2, NAXIS1).

VICAR indices are listed in First_Index_Fastest order, although this only affects the definitions of keywords N1, N2, N3, and N4. In practice, VICAR mixes together the issues of storage order and axis meaning, using other keywords NS, NL, NB, and ORG.

ISIS produces arrays in First_Index_Fastest storage order by default. ISIS has an output parameter switch that allows arrays to be produced in Last_Index_Fastest order.

4A.3 Array Storage Elements

PDS accommodates only binary data in arrays (no character formats). Characteristics of the homogeneous elements (or pixels) in an array must be described using the Element_Array class.

4A.4 Axis Meaning

In an N-dimensional array, each axis needs to be assigned a meaning. Common uses of axes are for spatial coordinates, RGB color, frequency or wavelength, and time.

PDS uses the terms *Line* and *Sample* to distinguish the two axes of a displayed array. It shares this usage with VICAR and ISIS, but not FITS. The relationship is defined exclusively by the storage order in the file, with the line number increasing more slowly than the sample number.

This means that, for Last_Index_Fastest ordering of a 2-D array, the first index would be the slower varying or the Line dimension, while the second index would be the faster varying or Sample dimension; the array would have index order (Line, Sample).

In an array with two spatial dimensions and one spectral dimension, PDS uses the term *Band* to designate the spectral axis. The array elements may be stored in the file in one of three possible storage orders. The index of the spectral axis may vary faster than those of both spatial axes, slower than both, or in between the two. An array in which the band (spectral) number varies slower than the line and sample numbers is typically referred to as *band sequential*. An array in which the band number varies faster than the line and sample numbers is typically identified as *sample interleaved* or *band interleaved by pixel*. An array in which the band number varies slower than the sample number, but faster than the line number, is referred to as *line interleaved* or *band interleaved by line*. For Last_Index_Fastest storage, a 3-D band sequential array (sometimes known as a 'cube') would have index order (Band, Line, Sample), a sample interleaved array would have index order (Line, Sample, Band), and a line interleaved array would have index order (Line, Band, Sample).

To maximize the efficiency of playing potentially large movies (Array_3D_Movie), the temporal axis should be the slowest varying axis.

Particle data often have one or more look directions (for example, polar and azimuthal angles) and an energy dimension. While these data have similarities to 'band sequential' formats, they are not image data in the traditional sense.

The characteristics of each array axis are modeled using the Axis_Array class. There must be one (and only one) Axis_Array class present in the label for each dimension of an array.

4A.5 Display Orientation

Properly defining the orientation in which an image should be presented on a display device is important when an array has two spatial dimensions. Each spatial axis of the array is associated with a direction on the display device. In most image display standards (including this one) the Sample, or faster varying axis, is associated with the horizontal direction on the display device. The Line, or slower varying axis, is associated with the vertical direction.

By almost universal convention, samples are displayed on devices from left to right. There is not, however, unanimity in how the line dimension should be handled. FITS, for example, assumes lines increase from bottom to top; VICAR and ISIS count lines from top to bottom. PDS leaves the choice of line display direction up to the data provider. This information (*e.g.*, down, up) is encoded in Array labels using the Display_Settings.Display_Direction class within the Discipline_Area of a label. These classes are in the Display dictionary – 'disp' namespace.

4B Table Base

The second of the four basic PDS4 structures is Table_Base. Data structures that consist of fixed-length rows of heterogeneous elements must be described using the Table_Base class or one of its subclasses.[5] At the current time, only two-dimensional tables are permitted.

Conceptually, tabular data consist of named columns containing data values at fixed locations. The data may consist of numbers and character strings including dates, times, and Boolean values; but any one column — also known as a field — contains only values of a single type.

Physically, the data are stored as a sequence of identically structured records where each record may be terminated by a record delimiter (required for Table_Character). Fields within each record are of fixed length and begin at fixed locations. Since both field lengths and record lengths are fixed, field values can be identified by position alone. However, field delimiters may optionally be included.

The data may be represented in binary, ASCII, or UTF-8. Binary tables must be described using the Table_Binary subclass of Table_Base, and ASCII and UTF-8 tables must be described using the Table_Character subclass. These two structures are similar; the former contains records described using a Record_Binary class and the latter contains records described using a Record_Character class which are followed by record delimiters. Because the overall structure and associations are very similar, only Table_Character is discussed in detail. An example Table_Character is below:

```
A-2  S After Deployment   2.3 0.9 7.9 1.2  7.4 0.7<CR><LF>
A-3  R Barnacle Bill      3.2 1.3 3.0 0.5 10.8 1.1<CR><LF>
A-4  S Next to Yogi       3.8 1.5 8.3 1.2  9.1 0.9<CR><LF>
A-5  S Dark Next to Yogi  2.8 1.1 7.5 1.1  8.7 0.9<CR><LF>
A-7  R Yogi               1.7 0.7 5.9 0.9  9.1 0.9<CR><LF>
```

The Record_Character class is used to describe the structure of each record in the table.

4B.1 Fields

There must be one explicitly defined Field_Character class for every field in a Record_Character except when Group_Field_Character can be used. Field definitions within a label must be in the same order as the physical appearance of the fields in the record.

4B.1.1 Field Length

Attribute <field_length> gives the number of bytes in a fixed-width field. Field delimiters (if any) and bracketing double quotes around character strings (if any) are not included in the count.

[5] See Section 4C.1 for information on table-like structures with variable length records.

4B.1.2 Field Formats

Attributes <field_format> and <validation_format> give the magnitude and precision of the data value.

<field_format> describes the general format of a value in a field.

For binary tables, this is used to format the value for output, particularly to specify the appropriate number of significant digits when converting real values.

For character tables, <field_format> is used to describe the maximum length and alignment of the data. <field_format> also gives an indication of the maximum precision of real numbers, but does not require all values to have this precision.

<validation_format> is used with character tables to further constrain a field's values. When a <validation_format> is present, the data in a table must match the format exactly in order to be valid. Additionally, if <validation_format> is present, it must not conflict with the field format. <validation_format> is not used for binary or delimited tables.

The specifier used in <field_format> and <validation_format> must be of an integer type if the field data type is a form of integer, or a floating point type if the field data type is a form of float, and of a string type for all other field data types.

***

The formation rule for a <field_format> or <validation_format> value is

```
%[+|-]width[.precision]specifier
```

where square brackets indicate an optional component, "%"is the percent sign (which must precede every field format value), and:

`[+|-]` denotes either a "+" or "-", but never both. The "-" may be used for string fields, to indicate that the string is (or should be) left-justified in the field. This is actually the preferred way to present most string values in character tables, so the <field_format> or <validation_format> value for fields with a data type of ASCII_String will nearly always begin with a "-". Similarly, the "-" denotes left justification for any of the date/time type fields. The "-" prefix is *forbidden* for all numeric fields (integers, floating point numbers, and numbers using scientific notation). The "+" may be used with numeric fields to indicate that an explicit sign is included in the field for input and should be displayed on output. In PDS4 labels, the "+" is *forbidden* for string fields.

`width` is the potential total width of the field — *i.e.*, the width of the widest value occurring in the field. `width` is an integer indicating the maximum number of characters needed for the complete representation of the largest (in terms of display bytes, not necessarily magnitude) value occuring, or potentially occuring, in the field. This should include bytes for signs, decimal points, and

exponents. In the case of string values, it is the maximum width from the first non-blank character to the last non-blank character. It does not include bytes for field delimiters or double quotes (“) around character strings, which are not considered part of the field. In character tables, it *must* be the same as <field_length> for scalar fields.

`width` is separated from `precision` by a decimal point ("."). If there is no `precision` specified, the decimal point must be omitted.

`precision` is the number of digits following the decimal point for real numbers (but is otherwise ignored). `precision` is used in three different ways:

1. For real numbers, it indicates the number of digits to the right of the decimal point.
2. For integers, it indicates that the integer will be zero-padded on the left out to the full field width. For example, the value "`2`" in "`%3.3d`" format is "`002`".
3. For strings, it signifies the maximum number of characters from the actual string value that should be printed. (It is possible in programming, for example, to print no more than the first 10 characters from a string, but require that the output field be left-justified and padded with at least 5 blanks by using a specifier of "`%15.10s`".) In PDS4 labels, if `precision` is included for a string format, it *must* be equal to `width`.

`specifier` is exactly one of the characters in the set `[doxfeEs]` where

- `d` indicates a decimal integer
- `o` indicates an unsigned octal integer
- `x` indicates an unsigned hexidecimal number
- `f` indicates a floating point number in the format *[-]ddd.ddd*, where the actual number of digits before and after the decimal point is determined by the preceding `width` and `precision` values (note that `width` includes the decimal point and any sign).
- `e,E` indicates a floating point number in the format *[-]d.ddde+/-dd* or *[-]d.dddE+/-dd* respectively where "+/-" stands for exactly one character (either "+" or "-"), there is always exactly one digit to the left of the decimal point, and the number of digits to the right of the decimal point is determined by the preceding `precision` value (note that the `width` includes all digits, signs, and the decimal point).
- `s` indicates a string value. Note that strings should generally be left-justified in fixed width character tables and on output from a binary

table, so most <field_format> values ending in "**s**" should begin with "-".

4B.2 Groups

Tables are constructed from records, and records are constructed from fields. Repeating sets of fields may be 'grouped' within a record, simplifying their definition.

The Group class defines the set of (repeating) fields and, possibly, (sub) groups. Required group attributes include:

<fields>	the number of scalar fields in the repeating structure
<groups>	the number of (sub)groups in the repeating structure
<repetitions>	the number of repetitions of the repeating structure

Fields and groups may be numbered at each level of nesting using the optional attributes

<field_number>	the position of a field within a series of fields, counting from 1
<group_number>	the position of a group within a series of groups, counting from 1

If two fields within a record are physically separated by one or more groups, they have consecutive field numbers; the fields within the intervening group(s) are numbered separately. Fields within a group separated by one or more (sub)groups, will also have consecutive field numbers. Similarly, group numbering is continuous across intervening fields.

4C Parsable Byte Stream

A parsable byte stream is a stream of bytes, either binary or character, that can be interpreted according to a standard set of rules. For example, a simple ASCII text file can be a parsable byte stream. It consists of a stream of character data; lines are delimited by a standard set of characters (usually the carriage-return line-feed pair). Many different applications are able to parse this format. An HTML file is also a parsable byte stream:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
           "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

The above file could be parsed and displayed by any browser programmed to understand the HTML 4.01 standard.

XML labels, comma separated value (CSV) tables, and SPICE kernels are other examples of parsable byte streams. Several formats of data headers are also recognized by PDS as parsable byte streams, including CDF, FITS, ISIS, ODL, TIFF, and VICAR headers as well as the PDS Header. Note that 'parsable' is not synonymous with 'human readable.'

The Parsable_Byte_Stream class and its subclasses are used to describe this form of data. The attribute <parsing_standard_id> is used to identify the parsing standard to be used.

4C.1 Delimiter Separated Value Format Description

The delimiter-separated value (DSV, or "spreadsheet") format has been used for some time in a variety of forms for storage and exchange of data between programs and systems. Special cases include the tab-separated value (TSV) and comma-separated value (CSV) formats. This section describes a general DSV format; it is based on RFC 4180 (http://tools.ietf.org/html/rfc4180).

The DSV format is defined as follows:

1. The data comprise one or more records.
2. Each record, including the last, is followed by a record delimiter, either the ASCII carriage-return line-feed pair (denoted `<CR><LF>`) or just the ASCII line-feed (`<LF>`). For example:

   ```
   aaa,bbb,ccc<CR><LF>
   zzz,yyy,xxx<CR><LF>
   ```

   <CR> and <LF> may be used only in combination as a record delimiter. They may not be used, either separately or in combination, as part of a field value. Whichever record delimiter is used (`<CR><LF>` or `<LF>`), it must be used consistently for all records.
3. Within each record, there will be one or more fields; fields are separated by field delimiters, and every record has the same number of fields. All field delimiters are the same. There is no field delimiter after the last field and before the record delimiter. The field delimiter must be one of the following characters: comma (`,`), semi-colon (`;`), vertical bar (`|`), or horizontal tab (`<HT>`). The first occurrence of one of these characters sets the delimiter to be used for all other fields. For example:

   ```
   aaa,bbb,ccc<LF>
   ```

   defines comma (`,`) as the delimiter; while:

   ```
   aaa|bbb|ccc<LF>
   aaa|b,b|ccc<LF>
   ```

   defines the vertical bar (`|`) to be the delimiter. Each record above contains three fields. The comma in the second record is part of the second field value.
4. A field may be empty. The interpretation of an empty field will be application and data type dependent. For example:

```
aaa,bbb,ccc<CR><LF>
aaa,,ccc<CR><LF>
```

has an empty field as the second field in the second record.

5. Leading and trailing spaces in a field are considered part of the field. For example:

```
aaa,bbb,   ccc<CR><LF>
zzz,yyy,xxxxxx<CR><LF>
```

contains three spaces in the third field of the first record, making its field length equal to 6. A field is the content between two delimiters, and a value is the information contained within a field. In most cases the field and value are synonymous; in this example, the application determines whether the value has 3 or 6 characters.

6. Fields may optionally be bracketed by a pair of bounding double quotes. The double quotes must be the first and last characters between delimiters; they are not counted in the field length and are not part of the field value. Any characters that are enclosed in double quotes are considered literal (including delimiters, and leading or trailing spaces); double quotes override the default specification of the field delimiter in 3 (above). If bracketing double quotes are used, there may be no double quotes within the field itself. In the example:

```
"aaa,bbb",ccc<LF>
```

the first comma is not treated as a delimiter; it is counted in the field length of the first field and is part of the corresponding value. The second comma in the record defines the field delimiter. This record consists of two fields (with values of aaa,bbb and ccc). In another example:

```
aaa,"   bbb",   ccc<LF>
```

the second and third fields consist of 6 characters each. The spaces in those two fields may be interpreted differently when the application extracts the values. The spaces in the second field are considered to be part of the value.

Double quote usage may vary from record to record. There is no requirement that a particular field be quoted in every record.

A pair of double quotes (,"",) is interpreted as an empty field (length 0).

4C.2 Delimited Tables

The previous section describes the format of delimited tables. This section explains how the characteristics of those tables must be described in labels.

The Table_Delimited class inherits attributes from the Parsable_Byte_Stream class, and it adds several more.

Each Table_Delimited class requires one Record_Delimited class, which describes the structure of all records in the delimited table.

Although the individual fields may vary in size from one record to the next, the number of fields, their names, and their data types must remain the same from line to line. There must be one Field_Delimited class present in the label to describe each field in the table record, except when Group_Field_Delimited can be used. Field definitions within the label must be in the same order as the physical appearance of the fields in the record.

The attribute <field_delimiter> must be defined in the Table_Delimited class (and it must be consistent with the provision in Section 4C.1 item 3). Attribute <maximum_field_length> gives the maximum number of bytes in a field. Field delimiters and bracketing double quotes around character strings (if any) are not included in the count. Values for attribute <field_format> are set as described in Section 4B.1.2.

Repeating sets of fields may be 'grouped' within a record, simplifying their definition, using the Group_Field_Delimited class as described in Section 4B.2.

4D Encoded Byte Stream

The encoded byte stream structure is a byte stream that may only be interpreted after it has been 'decoded' according to some well-known algorithm. For example, 'encoded' data may have been compressed and need to be uncompressed before interpretation. It is PDS policy that only publicly available, open source, widely accepted standards be used for the encoding of digital objects within the PDS; the attribute <encoding_standard_id> identifies the standard.

The subclasses of the Encoded Byte Stream class include Encoded_Binary (for files such as PDF and Microsoft Word documents), Encoded_Image (for browse, thumbnail, and document images stored in formats such as GIF, JPEG, and non-raster TIFF), Encoded_Audio (for ancillary and browse data in audio formats such as WAVE) and Encoded_Header (for binary headers such as TIFF headers on raster-formatted TIFF images). Encoded_Byte_Stream is not an option for Product_Observational.

5 Data Types

5A Attribute Value Types

Attribute value types are used to classify the data types of attribute values used in XML labels. These data types are typically specified in class and attribute descriptions in data dictionaries. For example, an ASCII_Integer value should contain only the characters 0-9, "+", and "-"; and an ASCII_String may contain blanks, carriage-returns, tabs, and other white space characters, which may or may not be removed, depending on the data type, before the value is interpreted by a subsequent application.

5A.1 Boolean Types

The PDS boolean data type is based on the primitive boolean type as defined in the World Wide Web Consortium (W3C) *XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* (W3C, 2012b). Permitted values are given in Table 5A-1.

Table 5A-1. Boolean Types

Data Type	Description	Permitted Values
ASCII_Boolean	True/False indicator	`true` and `false` (lower case only), or `1` (true) and `0` (false)

5A.2 Date and Time Types

PDS date and time formats are based on the extended formats of cardinal and ordinal date/time strings as defined in ISO 8601:2004 and as shown in Table 5A-2. Every PDS date- or time-related attribute and field must be defined using only one of the two possible formats. For example, the attributes <start_date_time> and <stop_date_time> are defined to be represented in calendar format. The two formats are represented as:

`YYYY-MM-DDThh:mm:ss.ffffffZ` (calendar format)

and

`YYYY-DOYThh:mm:ss.ffffffZ` (ordinal format)

where

`YYYY`	is the 4-digit year
`MM`	is the 2-digit month (possible values 01-12)
`DD`	is the 2-digit day of month (possible values (01-31)

`DOY`	is the 3-digit day of year (possible values 001-365, or 366 in a leap year)
`hh`	is the 2-digit hour (possible values 00-23)
`mm`	is the 2-digit minute (possible values (00-59)
`ss`	is the 2-digit second (possible values 00-59, or 60 if needed for a leap second)
`ffffff`	is decimal fractional seconds (one or more digits commensurate with precision)
`Z`	denotes UTC (may be required or optional depending on type).

- All components except `ffffff` and `Z` must be left padded with zeroes to reach the required number of digits.
- All times must be in the 24-hour clock format
- The ASCII period "." is the only delimiter allowed between the seconds and fractional seconds components.
- The ASCII colon is the only delimiter allowed between the other time components.
- The ASCII hypen "-" is the only delimiter allowed between date components.
- An upper case "T" is the only delimiter allowed between date and time components.
- Delimiters between components must be present if there are values on both sides; they must be omitted if there is not a value on each side.
- Precision may be reduced by dropping any delimiter and all components to its right.
- A time only format may be obtained by dropping "T" and all components to its left.
- Any value not specifically indicated as being UTC by including the "Z" indicator is assumed to be a local time.

Any data preparer who anticipates archiving data with dates earlier than 15 October 1582 or later than 31 December 9999 must consult PDS for special instructions.

Native times, such as from spacecraft clock counters, have formats defined in the appropriate mission or discipline dictionary.

All values for the attributes <start_date_time> and <stop_date_time> must be given in UTC.

The optional variable length decimal fraction of seconds is denoted by `[.ffffff]`. `[Z]` indicates that the suffix is optional, but PDS strongly encourages its use when the time is given in UTC.

Table 5A-2. Date and Time Types		
Data Type	**Description**	**Permitted Values**

ASCII_Date_DOY	An ASCII date string in ordinal format.	Date value in either of the following forms:
		`YYYY[Z]`
		`YYYY-DOY[Z]`

ASCII_Date_YMD	An ASCII date string in calendar format.	Date value in any of the following forms:
		`YYYY[Z]`
		`YYYY-MM[Z]`
		`YYYY-MM-DD[Z]`

ASCII_Date_Time_DOY_UTC	An ASCII date/time string in ordinal format that must be in Coordinated Universal Time (UTC).	Date/time value in any of the following forms:[6]
		`YYYY-DOYThhZ`
		`YYYY-DOYThh:mmZ`
		`YYYY-DOYThh:mm:ss[.ffffff]Z`

ASCII_Date_Time_YMD_UTC	An ASCII date/time string in calendar format that must be in Coordinated Universal Time (UTC).	Date/time value in any of the following forms: [7]
		`YYYY-MM-DDThhZ`
		`YYYY-MM-DDThh:mmZ`
		`YYYY-MM-DDThh:mm:ss[.ffffff]Z`

ASCII_Date_Time_DOY	An ASCII date/time string in ordinal format.	Date/time value in any of the following forms: [7]
		`YYYY-DOYThh[Z]`
		`YYYY-DOYThh:mm[Z]`
		`YYYY-DOYThh:mm:ss[.ffffff][Z]`

ASCII_Date_Time_YMD	An ASCII date/time string in calendar format.	Date/time value in any of the following forms: [7]
		`YYYY-MM-DDThh[Z]`
		`YYYY-MM-DDThh:mm[Z]`
		`YYYY-MM-DDThh:mm:ss[.ffffff][Z]`

[6] In cases when time information is not available, this format may be truncated to a date string of as little as the year (YYYY).

Table 5A-2. Date and Time Types		
Data Type	**Description**	**Permitted Values**

ASCII_Time	An ASCII time string. May be used for local times on Earth or local solar time on other planets.	Time value in any of the following forms:
		`hh[Z]`
		`hh:mm[Z]`
		`hh:mm:ss[.ffffff][Z]`

5A.3 Numeric Types

PDS numeric data types are based on a mixture of primitive and derived types defined in W3C, 2012b. The specific base type for each data type is indicated in the Description column in Table 5A-3.

Table 5A-3. Numeric Types		
Data Type	**Description**	**Permitted Values**
ASCII_Integer	An ASCII character representation of a signed 64-bit integer in base 10. Based on the derived data type `xs:long`.	An ASCII string consisting of the digits `0` through `9`, optionally prefixed with a positive "+" or negative "–" sign. Values must be within the range -2^63 to 2^63-1, inclusive.
ASCII_NonNegative_Integer	An ASCII character representation of an unsigned 64-bit integer in base 10. Based on the derived data type `xs:unsignedLong`.	An ASCII string consisting of the digits `0` through `9`. Values must be within the range 0 to 2^64-1, inclusive.

Table 5A-3. Numeric Types		
Data Type	**Description**	**Permitted Values**
ASCII_Real	An ASCII character representation of an IEEE 754 64-bit floating point number. Based on the primitive data type `xs:double`.	An ASCII string consisting of: An optional sign, consisiting of '+' or '-', A mantissa, consisting of one of the following: • A series of digits from 0-9. • A series of digits from 0-9, a decimal point, and an optional series of digits from 0-9. • A decimal point and a series of digits from 0-9. An optional exponent, consisting of: • The letter 'e' or 'E', • An optional sign, consisiting or '+' or '-' • A series of digits from 0-9 Values must fit into the value space of an IEEE 754-2008 binary64 number. PDS does not allow positive infinity (INF), negative infinity (-INF) or not-a- number (NaN).
ASCII_Numeric_Base2	An ASCII character representation of a non-negative unsigned integer in base 2. This is a PDS defined data type.	An ASCII string consisting of the characters `0` and `1`. May not be preceded by any sign (+/-) notation. Limited to 255 characters.
ASCII_Numeric_Base8	An ASCII character representation of a non-negative unsigned integer in base 8. This is a PDS defined data type.	An ASCII string consisting of the digits `0` through `7`. May not be preceded by any sign (+/-) notation. Limited to 255 characters.

Table 5A-3. Numeric Types		
Data Type	**Description**	**Permitted Values**
ASCII_Numeric_Base16	An ASCII character representation of a non-negative unsigned integer in base 16. Based on the primitive data type `xs:hexBinary`.	An ASCII string consisting of the characters `0` through `9` and `A` through `F` or `a` through `f`. May not be preceded by any sign (+/-) notation. Limited to 255 characters.
ASCII_MD5_Checksum	An ASCII representation of a 128-bit hash value calculated using the MD5 algorithm (RFC 1321). This is a PDS defined data type.	An ASCII string consisting of the characters `0` through `9` and `A` through `F` or `a` through `f`. Must be exactly 32 characters in length.

5A.4 String Types

PDS string data types are based on a mixture of primitive and derived types defined in W3C, 2012b and PDS types derived from these. The specific base type for each data type is indicated in the Description column in Table 5A-4.

Table 5A-4. String Types		
Data Type	**Description**	**Permitted Values**
ASCII_AnyURI	A URI, and its subclasses URN and URL. `xs:anyURI`.	See W3C, 2012b
ASCII_BibCode	A bibliographic code as assigned by the Astrophysics Data System	See Astrophysics Data System documentation
ASCII_DOI	A Digital Object Identifier (DOI) as assigned by members of the International DOI Federation.	See International DOI Federation documentation
ASCII_LID	A PDS logical identifier.	See Section 6D.2
ASCII_LIDVID	A PDS versioned identifier (logical identifier plus version identifier).	See Section 6D.3

Table 5A-4. String Types		
Data Type	**Description**	**Permitted Values**
ASCII_LIDVID_LID	Either a PDS logical identifier or PDS versioned identifier.	See Sections 6D.2 and 6D.3
ASCII_VID	A PDS version identifier	See Section 6D.3
ASCII_Directory_Path_Name	A directory path in UNIX format.	ASCII string of the form dir1/dir2/. See Section 6C.2.
ASCII_File_Name	A file name.	ASCII string of the form filename.ext. See Section 6C.1.
ASCII_File_Specification_Name	A directory path and file name (including file name extension) in UNIX format.	ASCII string of the form dir1/dir2/filename.ext. See Section 6C.
ASCII_Short_String_Collapsed	An ASCII-encoded text string with white space collapsed — that is, contiguous spaces, line feeds, tabs, and carriage returns have been collapsed into a single ASCII space character and leading and trailing spaces have been removed.	An ASCII string containing no more than 255 characters and including no leading or trailing <SP>, no more than one contiguous <SP>, no <LF>, no <HT>, and no <CR>.
ASCII_Short_String_Preserved	An ASCII-encoded text string with all characters preserved.	An ASCII string of no more than 255 characters.
ASCII_Text_Collapsed	An ASCII-encoded text string of unlimited length with white space collapsed — that is, contiguous spaces, line feeds, tabs, and carriage returns have been collapsed into a single ASCII space character and leading and trailing spaces have been removed.	An ASCII string containing no leading or trailing <SP>, no more than one contiguous <SP>, no <LF>, no <HT>, and no <CR>.
ASCII_Text_Preserved	An ASCII-encoded text string of unlimited length with all characters preserved.	An ASCII string.

Table 5A-4. String Types		
Data Type	**Description**	**Permitted Values**
UTF8_Short_String_Collapsed	A UTF-8 encoded text string with white space collapsed (see ASCII_Short_String_Collapsed above).	A UTF-8 string containing no more than 255 bytes and including no leading or trailing <SP>, no more than one contiguous <SP>, no <LF>, no <HT>, and no <CR>.
UTF8_Short_String_Preserved	A UTF-8 encoded text string with all characters preserved.	A UTF-8 string of no more than 255 bytes.
UTF8_Text_Collapsed	A UTF-8 encoded text string with white space collapsed (see ASCII_Short_String_Collapsed above).	A UTF-8 string including no leading or trailing <SP>, no more than one contiguous <SP>, no <LF>, no <HT>, and no <CR>.
UTF8_Text_Preserved	A UTF-8 encoded text string with all characters preserved.	A UTF-8 string.

5B Character Data Types

Character data types are used to describe the data formats of character fields in tables.

Table character fields should use the data types described in Table 5B-1. The values described in Section 5A for Boolean Types (Section 5A.1), Date and Time Types (Section 5A.2), and Numeric Types (Section 5A.3) may also be used for field descriptions in tables.

Table 5B-1. Character Data Types

Data Types	Description	Permitted Values
ASCII_AnyURI	A URI, and its subclasses URN and URL. `xs:anyURI`	See W3C, 2012b
ASCII_BibCode	A bibliographic code as assigned by the Astrophysics Data System.	See Astrophysics Data System documentation.
ASCII_DOI	A Digital Object Identifier (DOI) as assigned by members of the International DOI Federation.	See International DOI Federation documentation.
ASCII_File_Name	A file name.	ASCII string of the form: filename.ext. See Section 6C.1.
ASCII_File_Specification_Name	A directory path and file name (including file name extension) in UNIX format.	ASCII string of the form: dir1/dir2/filename.ext. See Section 6C.
ASCII_LID	A PDS logical identifier.	See Section 6D.2.
ASCII_LIDVID	A PDS versioned identifier (logical identifier plus version identifier).	See Section 6D.3.
ASCII_VID	A PDS version identifier.	See Section 6D.3.
ASCII_String	An ASCII string.	An ASCII string.
UTF8_String	A UTF-8 string.	A UTF-8 string.

5C Binary Data Types

Binary data types are used to describe data formats of fields in binary tables and array elements in arrays.

5C.1 Integers

5C.1.1 Signed LSB Integers

This section describes signed integers stored in Least Significant Byte (LSB) first (also known as 'little-endian') order. In this section the following definitions apply:

B1 – B8 Arrangement of bytes as they appear when reading a file — that is, read byte B1 first, then B2, B3 and B4, up through B8.

i-sign Integer sign bit – bit 1 in the highest order byte

I1 – I8 Arrangement of bytes in the integer, from lowest order (I1) to highest order (I8). The bits within each byte are counted from left to right (b1 to b8) but interpreted from right to left (*i.e.*, lowest value = bit 8, highest value = bit 1), in the following way:

8-byte integers (Fig. 5C-1):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

In I5, bits b1 to b8 represent 2^{39} through 2^{32}, respectively

In I6, bits b1 to b8 represent 2^{47} through 2^{40}, respectively

In I7, bits b1 to b8 represent 2^{55} through 2^{48}, respectively

In I8, bits b2 to b8 represent 2^{62} through 2^{56}, respectively

4-byte integers (Fig. 5C-2):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b2 to b8 represent 2^{30} through 2^{24}, respectively

2-byte integers (Fig. 5C-3):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b2 to b8 represent 2^{14} through 2^8, respectively

1-byte integer (Fig. 5C-4):

In I1, bits b2 to b8 represent 2^6 through 2^0, respectively

All negative values are represented in two's complement.

data_type = SignedLSB8

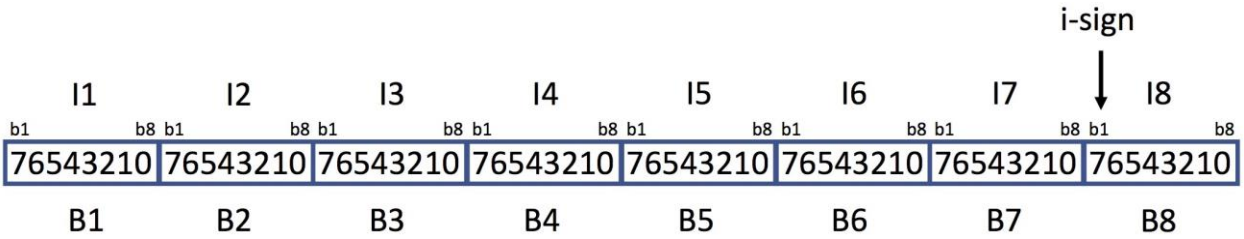


Fig. 5C-1. Signed 8-byte integer, least significant byte (I1) first

data_type = SignedLSB4

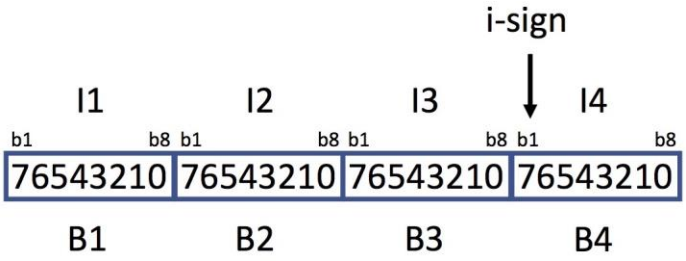


Fig. 5C-2. Signed 4-byte integer, least significant byte (I1) first

data_type = SignedLSB2

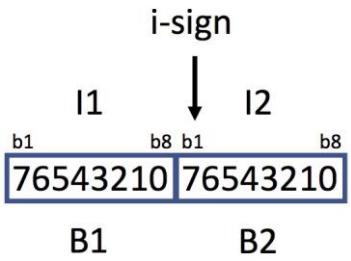


Fig. 5C-3. Signed 2-byte integer, least significant byte (I1) first

data_type = SignedByte

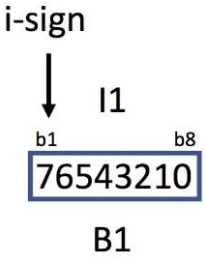


Fig. 5C-4. Signed 1-byte integer

5C.1.2 Unsigned LSB Integers

This section describes unsigned integers stored in LSB format. In this section the following definitions apply:

B1 – B8 Arrangement of bytes as they appear when reading a file — that is, read byte B1 first, then B2, B3 and B4, up through B8.

I1 – I8 Arrangement of bytes in the integer, from lowest order (I1) to highest order (I8). The bits within each byte are counted from left to right (b1 to b8) but interpreted from right to left (*i.e.*, lowest value = bit 8, highest value = bit 1), in the following way:

8-byte integers (Fig. 5C-5):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

In I5, bits b1 to b8 represent 2^{39} through 2^{32}, respectively

In I6, bits b1 to b8 represent 2^{47} through 2^{40}, respectively

In I7, bits b1 to b8 represent 2^{55} through 2^{48}, respectively

In I8, bits b1 to b8 represent 2^{63} through 2^{56}, respectively

4-byte integers (Fig. 5C-6):

In I1, bits b1 to b8 represent 2^{7} through 2^{0}, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^{8}, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

2-byte integers (Fig. 5C-7):

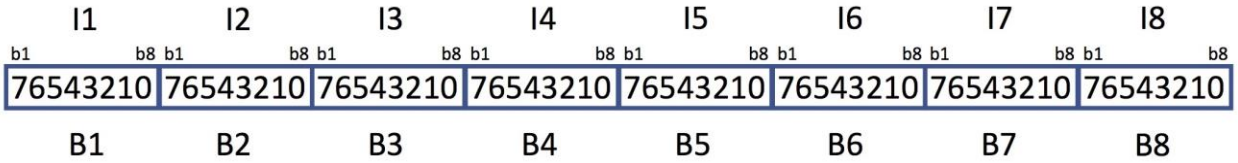
In I1, bits b1 to b8 represent 2^{7} through 2^{0}, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^{8}, respectively

1-byte integers (Fig. 5C-8):

In I1, bits b1 to b8 represent 2^{7} through 2^{0}, respectively

data_type = UnsignedLSB8



Fig, 5C-5. Unsigned 8-byte integer, least significant byte (I1) first.

data_type = UnsignedLSB4

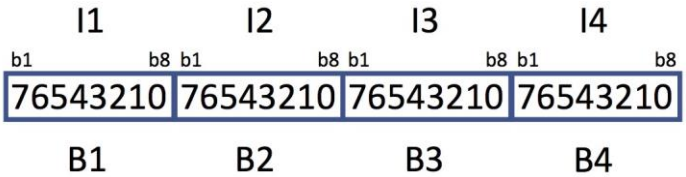


Fig. 5C-6. Unsigned 4-byte integer, least significant byte (I1) first.

data_type = UnsignedLSB2

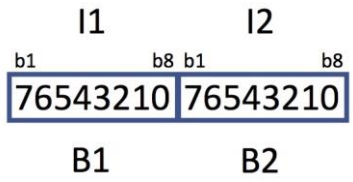


Fig. 5C-7. Unsigned 2-byte integer, less significant byte(I1) first.

data_type = UnsignedByte

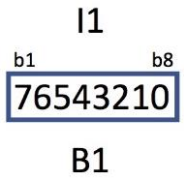


Fig. 5C-8. Unsigned 1-byte integer

5C.1.3 Signed MSB Integers

This section describes the signed integers stored in Most Significant Byte (MSB) first (also known as 'big-endian') order. In this section the following definitions apply:

B1 – B8 Arrangement of bytes as they appear when read from a file — that is, read B1 first, then B2, B3, and B4, up through B8.

i-sign Integer sign bit – bit 1 in the highest order byte

I1 – I8 Arrangement of bytes in the integer, from lowest order (I1) to highest order (I8). The bits within each byte are counted from left to right (b1 to b8) but interpreted from right to left (*i.e.*, lowest value = bit 8, highest value = bit 1) in the following way:

8-byte integers (Fig. 5C-9):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

In I5, bits b1 to b8 represent 2^{39} through 2^{32}, respectively

In I6, bits b1 to b8 represent 2^{47} through 2^{40}, respectively

In I7, bits b1 to b8 represent 2^{55} through 2^{48}, respectively

In I8, bits b2 to b8 represent 2^{62} through 2^{56}, respectively

4-byte integers (Fig. 5C-10):

In I1, bits b1 to b8 represent 2^{7} through 2^{0}, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^{8}, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b2 to b8 represent 2^{30} through 2^{24}, respectively

2-byte integers (Fig. 5C-11):

In I1, bits b1 to b8 represent 2^{7} through 2^{0}, respectively

In I2, bits b2 to b8 represent 2^{14} through 2^{8}, respectively

1-byte integers (Fig. 5C-4):

In I1, bits b2 to b8 represent 2^{6} through 2^{0}, respectively

All negative values are represented in two's complement.

data_type = SignedMSB8

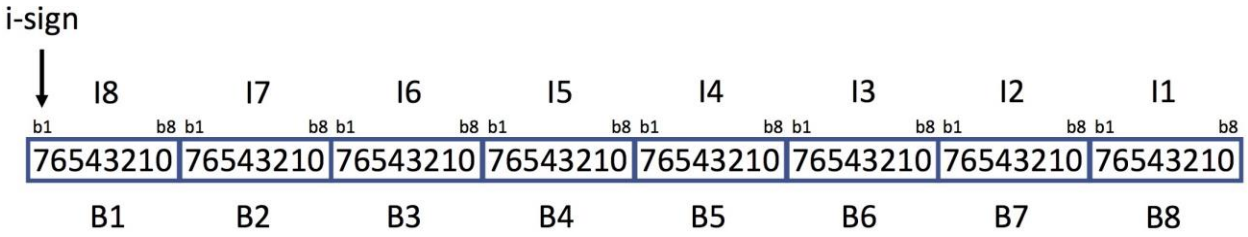


Fig. 5C-9. Signed 8-byte integer, most significant byte (I8) first.

data_type = SignedMSB4

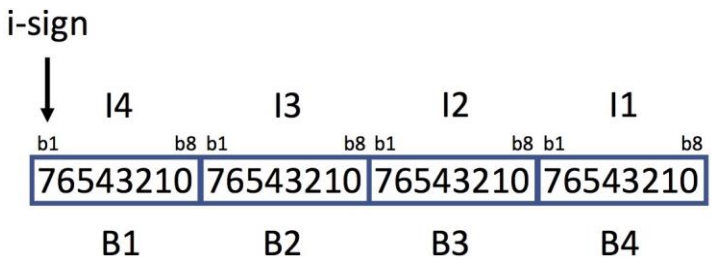


Fig. 5C-10. Signed 4-byte integer, most significant byte (I4) first.

data_type = SignedMSB2

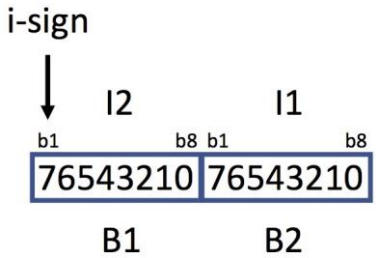


Fig. 5C-11. Signed 2-byte integer, more significant byte (I2) first

The signed 1-byte integer is shown in Figure 5C-4.

5C.1.4 Unsigned MSB Integers

This section describes unsigned integers stored in MSB format. In this section the following definitions apply:

B1 – B8 Arrangement of bytes as they appear when read from a file — that is, read B1 first, then B2, B3, and B4, up through B8)

I1 – I8 Arrangement of bytes in the integer, from lowest order (I1) to highest order (I8). The bits within each byte are counted from left to right (b1 to b8) but interpreted from right to left (*i.e.*, lowest value = bit 8, highest value = bit 1) in the following way:

8-byte integers (Fig. 5C-12):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

In I5, bits b1 to b8 represent 2^{39} through 2^{32}, respectively

In I6, bits b1 to b8 represent 2^{47} through 2^{40}, respectively

In I7, bits b1 to b8 represent 2^{55} through 2^{48}, respectively

In I8, bits b1 to b8 represent 2^{63} through 2^{56}, respectively

4-byte integers (Fig. 5C-13):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

In I3, bits b1 to b8 represent 2^{23} through 2^{16}, respectively

In I4, bits b1 to b8 represent 2^{31} through 2^{24}, respectively

2-byte integers (Fig. 5C-14):

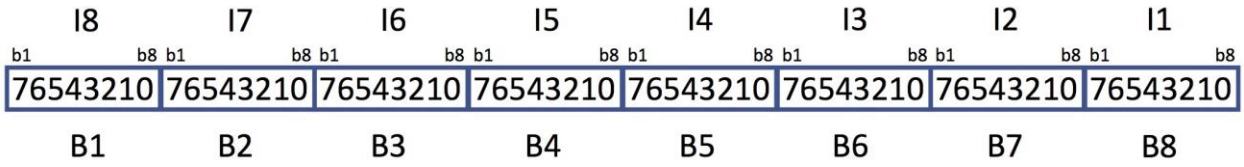
In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

In I2, bits b1 to b8 represent 2^{15} through 2^8, respectively

1-byte integers (Fig. 5C-8):

In I1, bits b1 to b8 represent 2^7 through 2^0, respectively

data_type = UnsignedMSB8



Fig, 5C-12. Unsigned 8-byte integer, most significant byte (I8) first.

data_type = UnsignedMSB4

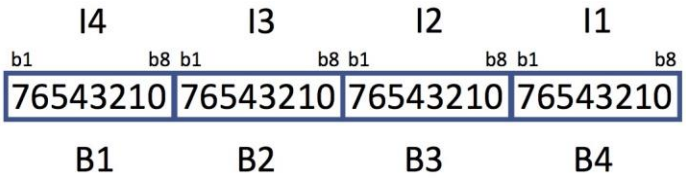


Fig. 5C-13. Unsigned 4-byte integer, most significant byte (I4) first

data_type = UnsignedMSB2

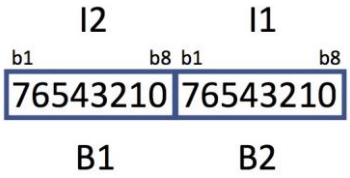


Fig. 5C-14. Unsigned 2-byte integer, more significant byte (I2) first

The unsigned 1-byte integer is shown in Figure 5C-8.

5C.2 Reals

This section describes the internal IEEE floating-point representation of real numbers. In this section the following definitions apply:

B1 – B8 Arrangement of bytes as they appear when read from a file — that is, read B1 first, then B2, B3, and B4 up through B8.

m-sign Mantissa sign bit

E1 –E2 Arrangement of the portions of the bytes that make up the exponent, from lower order (E1) to higher order (E2). The bits within each byte are counted from left to right (b1 to b8) but interpreted from right to left (*i.e.*, lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

8-byte double precision (see Figure 5C-15 for LSB and Figure 5C-17 for MSB):

In E1, bits b1 to b4 represent 2^3 through 2^0, respectively

In E2, bits b2 to b8 represent 2^{10} through 2^4, respectively

Exponent bias = 1023

4-byte single precision (see Figure 5C-16 for LSB and Figure 5C-18 for MSB):

In E1, bit b1 represents 2^0

In E2, bits b2 to b8 represent 2^7 through 2^1, respectively

Exponent bias = 127

M1 – M7 Arrangement of the portions of the bytes that make up the mantissa, from lowest order fraction (M1) to the highest order fraction (M7). The bits within each bytes are counted left to right (b1 to b8) but interpreted as inverse fractional powers of two from right to left with the rightmost bit having the smallest value, in the following way:

8-byte double precision (see Figure 5C-15 for LSB and Figure 5C-17 for MSB):

In M1, bits b5 to b8 represent $1/2^1$ through $1/2^4$, respectively

In M2, bits b1 to b8 represent $1/2^5$ through $1/2^{12}$, respectively

In M3, bits b1 to b8 represent $1/2^{13}$ through $1/2^{20}$, respectively

In M4, bits b1 to b8 represent $1/2^{21}$ through $1/2^{28}$, respectively

In M5, bits b1 to b8 represent $1/2^{29}$ through $1/2^{36}$, respectively

In M6, bits b1 to b8 represent $1/2^{37}$ through $1/2^{44}$, respectively

In M7, bits b1 to b8 represent $1/2^{45}$ through $1/2^{52}$, respectively

4-byte single precision (see Figure 5C-16 for LSB and Figure 5C-18 for MSB):

In M1, bits b2 to b8 represent $1/2^1$ through $1/2^7$, respectively

In M2, bits b1 to b8 represent $1/2^8$ through $1/2^{15}$, respectively

In M3, bits b1 to b8 represent $1/2^{16}$ through $1/2^{23}$, respectively

The stored value may be recovered as follows:

$$1.\mathit{mantissa} \times 2^{(\mathit{exponent} - \mathit{bias})}$$

Note that the integer part ("1.") is implicit in all formats as described above. In all cases the exponent is stored as an unsigned, biased integer (that is, the stored exponent value - bias value = true exponent).

data_type = IEEE754LSBDouble

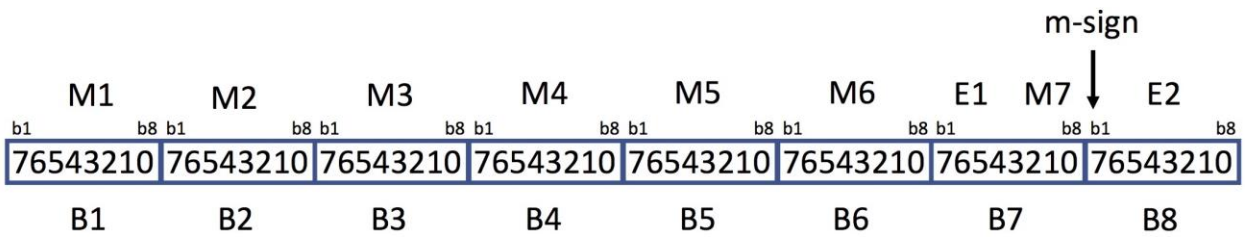


Fig. 5C-15. IEEE 754 double precision real, least significant byte (M1) first.

data_type = IEEE754LSBSingle

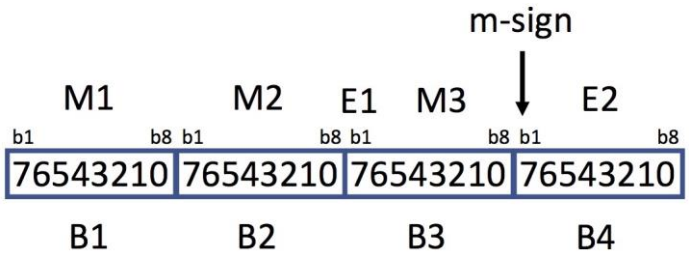


Fig. 5C-16. IEEE 754 single precision real, least significant byte (M1) first.

data_type = IEEE754MSBDouble

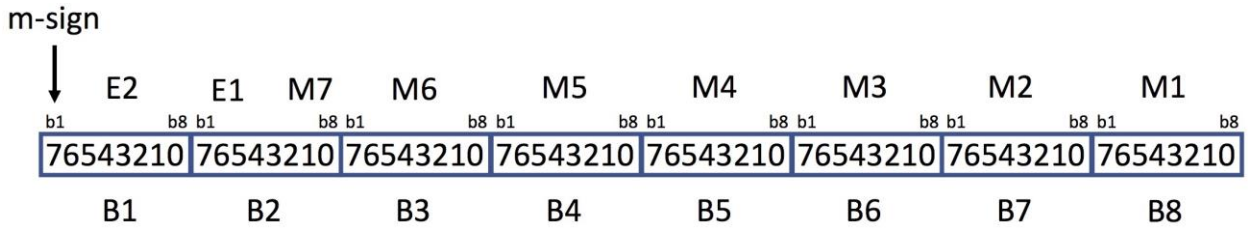


Fig. 5C-17. IEEE 754 double precision real, most significant byte (m-sign plus E2) first.

data_type = IEEE754MSBSingle

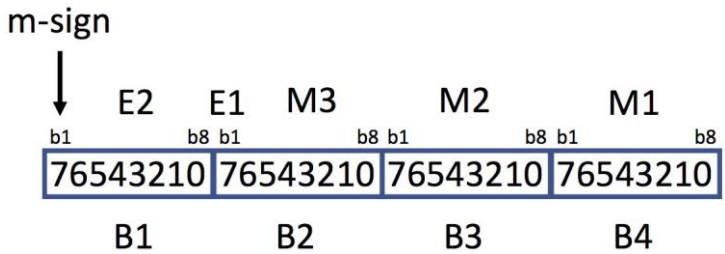


Fig. 5C-18. IEEE 754 single precision real, most significant byte (M-sign plus E2) first

5C.3 Complex

This section describes the internal IEEE floating-point representation of complex numbers, illustrated in Figures 5C-19, 5C-20, 5C-21, and 5C-22. IEEE complex numbers consist of two IEEE-format real numbers (Section 5C.2) of the same precision, which are contiguous in memory. The first number represents the real part and the second number represents the imaginary part of the complex value.

data_type = ComplexLSB16

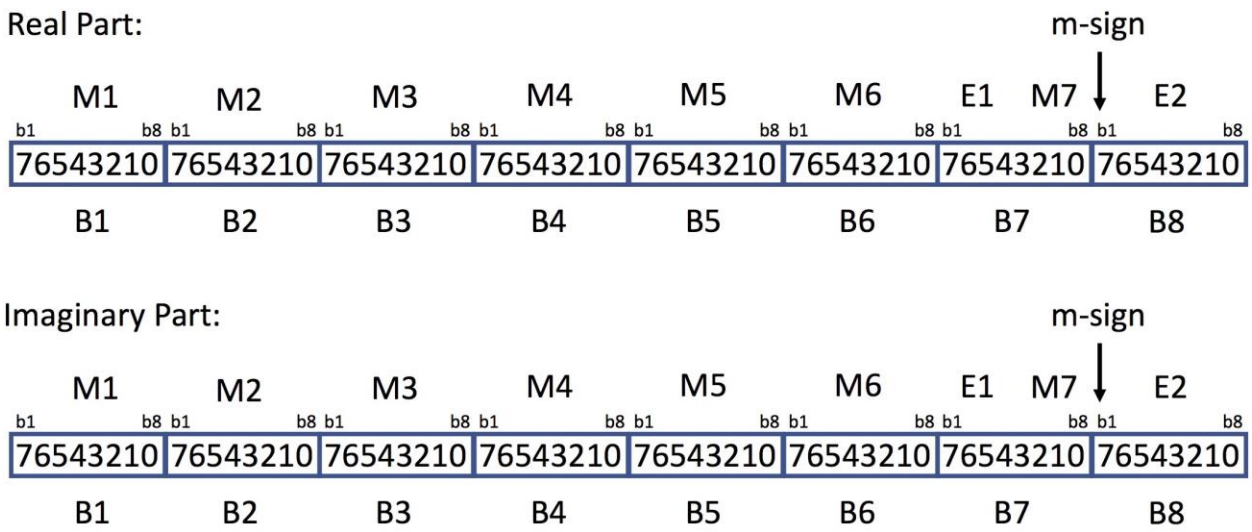


Fig. 5C-19. Double precision complex, least significant bytes (M1) first; the real component is first, the imaginary component second.

data_type = ComplexLSB8

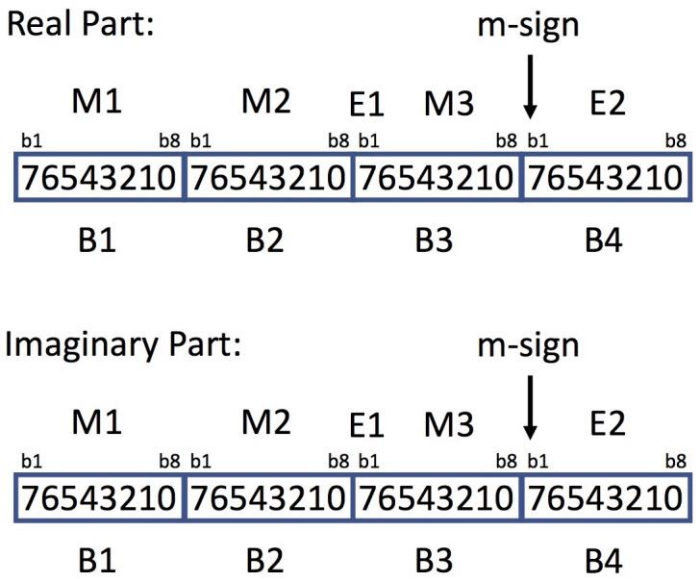


Fig. 5C-20. Single precision complex, least significant bytes (M1) first.

data_type = ComplexMSB16

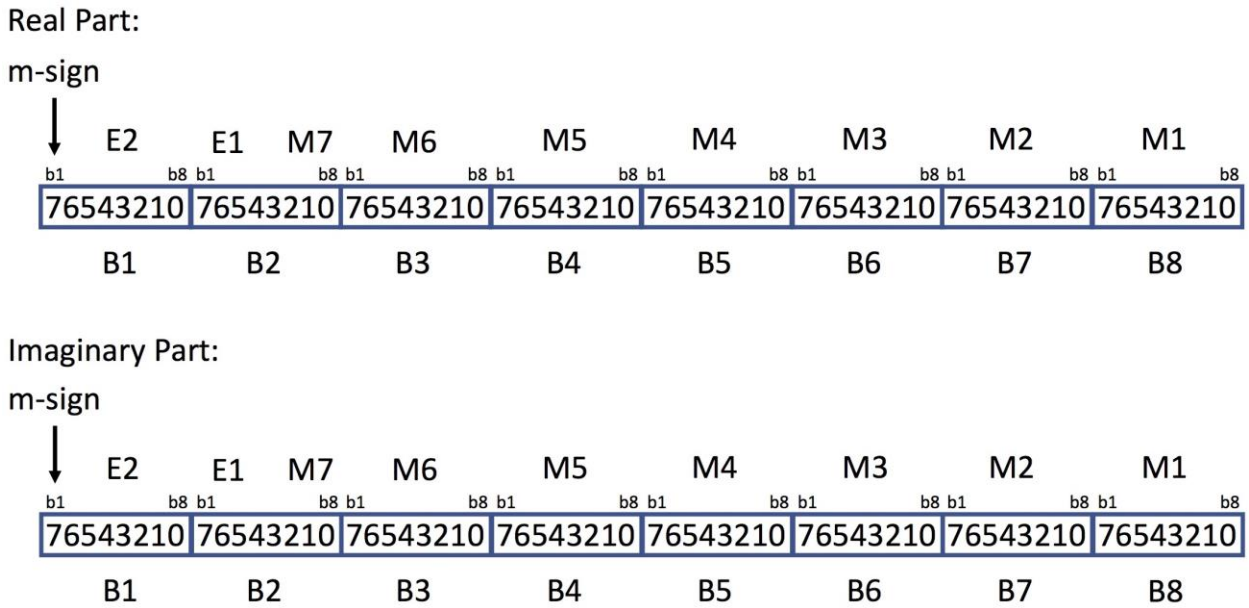


Fig. 5C-21. Double precision complex, most significant bytes (m-sign plus E2) first.

data_type = ComplexMSB8

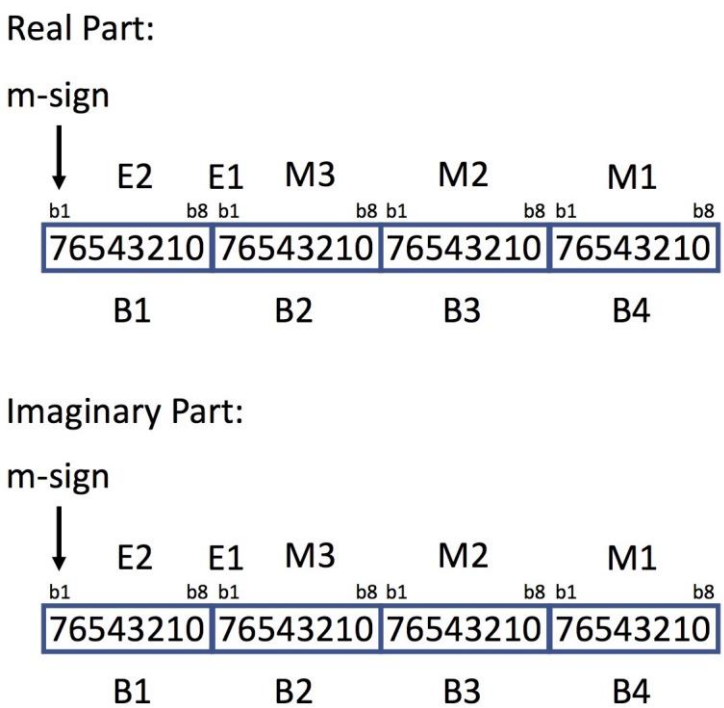


Fig. 5C-22. Single precision complex, most significant bytes (m-sign plus E2) first.

5C.4 Bit Strings

Bit strings are sequences of contiguous bits extending over one or more sequential bytes[7]; PDS places no general constraints on the location or length of bit strings. However, there are restrictions on their use in observational data — see the PDS Policy on Packed Data (https://pds.nasa.gov/datastandards/documents/policy/PolicyOnPackedData04192017.pdf).

In the examples below, B1 is the first byte read and B2 is the second.

Bits are numbered left to right within each byte (b1 to b8). When the bit string covers more than one byte, the numbering may be extended (in the examples below the numbering could extend from b1 to b16 rather than the two b1 to b8 ranges shown).

The remainder of this section describes signed and unsigned integers stored as bit strings. Such bit strings are limited to 64 bits each.

5C.4.1 Unsigned Integers Stored as Bit Strings

Significance of the bits (that is, their values) increases from right to left within the bit string — from 2^0 at the extreme right to 2^{n-1} at the extreme left, where n is the number of bits in the string.

[7] To ensure bit contiguity across boundaries, bytes are read sequentially.

Fig. 5C-23 shows an example unsigned bit string that extends from bit b6 of B1 to bit b2 of B2 (five bits total).

In B1, bits b6 to b8 represent 2^4, 2^3, and 2^2, respectively.
In B2, bits b1 and b2 represent 2^1 and 2^0, respectively

data_type = UnsignedBitString

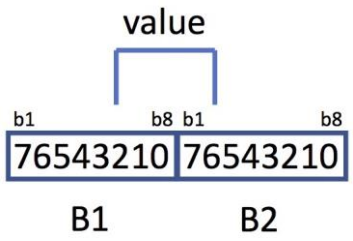


Fig. 5C-23. Unsigned bit string

5C.4.2 Signed Integers Stored as Bit Strings

Fig. 5C-24 shows an example signed bit string extending from bit b5 of B1 to bit b2 of B2 (6 bits total). The sign bit is bit 5 of B1. If the sign bit is "0" the value is positive, if "1", the value is negative, and it is represented in two's complement notation.

In B1, bits b6 to b8 represent 2^4, 2^3, and 2^2, respectively.
In B2, bits b1 and b2 represent 2^1 and 2^0, respectively.

data_type = SignedBitString

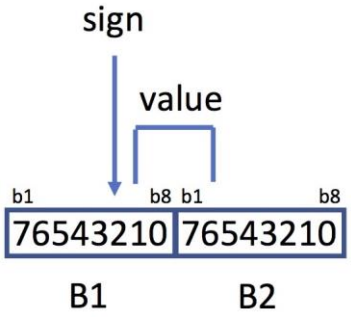


Fig. 5C-24. Signed bit string

6 Naming

6A Character Sets

PDS recognizes several character sets.

6A.1 ASCII Character Set (also known as IsBasicLatin)

The ASCII character set is the encoding of characters into 7-bit integer numbers according to the ANSI standard (see Section 1.5):

Octal - Character

000 NUL	001 SOH	002 STX	003 ETX	004 EOT	005 ENQ	006 ACK	007 BEL
010 BS	011 HT	012 NL	013 VT	014 NP	015 CR	016 SO	117 SI
020 DLE	021 DC1	022 DC2	023 DC3	024 DC4	025 NAK	026 SYN	027 ETB
030 CAN	031 EM	032 SUB	033 ESC	034 FS	035 GS	036 RS	037 US
040 SP	041 !	042 "	043 #	044 $	045 %	046 &	047 '
050 (	051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [	134 \	135]	136 ^	137 _
140 `	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174 |	175 }	176 ~	177 DEL

Hexadecimal - Character

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /

30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL

Decimal - Character

0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL
8 BS	9 HT	10 NL	11 VT	12 NP	13 CR	14 SO	15 SI
16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB
24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
32 SP	33 !	34 "	35 #	36 $	37 %	38 &	39 '
40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

where the following are non-printing, control characters:

```
NUL = null character
SOH = start of header
STX = start of text
ETX = end of text
EOT = end of transmission
```

```
ENQ = enquiry
ACK = acknowledgement
BEL = bell
BS  = backspace
HT  = horizontal tab
LF  = line feed
VT  = vertical tab
FF  = form feed
CR  = carriage return
SO  = shift out
SI  = shift in
DLE = data link escape
DC1 = device 1 control
DC2 = device 2 control
DC3 = device 3 control
DC4 = device 4 control
NAK = negative acknowledgement
SYN = synchronous idle
ETB = end transmission block
CAN = cancel
EM  = end of medium
SUB = substitute
ESC = escape
FS  = file separator
GS  = group separator
RS  = record separator
US  = unit separator
DEL = delete
```

6A.2 ASCII Alphanumeric Character Set

The ASCII alphanumeric character set includes the upper and lower case ASCII letters and the ASCII digits.

upper case letters	A-Z	ASCII 0x41 to 0x5a
lower case letters	a-z	ASCII 0x61 to 0x7a
digits	0-9	ASCII 0x30 to 0x39

6A.3 ASCII Printable Character Set

The ASCII printable character set is the set in Section 6A.1 less the non-printing, control characters. The hexadecimal equivalents are 0x20 to 0x7E, inclusive.

6A.4 UTF-8

UTF-8 is a variable-width encoding scheme that represents the Unicode character set; it was constrained by RFC 3629 to four bytes in November 2003. UTF-8 is backwardly compatible with the ASCII character set while its 2-, 3-, and 4-byte options permit encoding of virtually any character used worldwide. For more information, see the following:

RFC 3629	http://www.tools.ietf.org/html/rfc3629
UTF-8	http://en.wikipedia.org/wiki/UTF-8
Unicode	http://www.unicode.org/versions/Unicode6.2.0/

6B Namespace

A *namespace* is a context within which attributes and classes may be defined; it is managed by a *steward*. A namespace may have only one steward, but a steward is allowed to manage more than one namespace. PDS is a *registration authority*; it oversees several namespaces, each closely controlled by a steward; the PDS Engineering Node (EN), acting on behalf of the PDS registration authority, coordinates the stewards.

There is a common namespace managed by EN (under the steward name 'pds')[8]; each PDS discipline node also manages a namespace. Typically, each planetary mission has a namespace; additionally, some areas of study ('disciplines', such as cartography) have namespaces. Only namespaces that are registered in the Namespace Registry (https://pds.nasa.gov/datastandards/schema/pds-namespace-registry.pdf) are allowed in PDS labels.

Table 6B-1 shows example namespaces recognized by PDS, along with their Uniform Resource Identifiers (URIs) and their stewards.

6B.1 Namespace Creation and Use

To ensure that namespaces are unique across PDS, EN creates them in consultation with the discipline nodes or with the corresponding international archiving authority. Typically namespaces are created for new investigations (such as missions) or science specialties. When circumstances appear to warrant the creation of a new namespace, a data provider should confer with the consulting discipline node. Discipline nodes should also be contacted when there is a question about which namespace to use or how to reference a namespace.

---

[8] An exception to this rule is the attribute Instrument.Subtype, which is in the common namespace, but which is defined and managed by the appropriate discipline node.

6B.2 Formation of Namespace IDs:

The following rules govern creation of namespace identifiers; namespace_id values must be:

- unique across the PDS.
- no longer than 16 characters.
- constructed from ASCII lower case letters, digits, and dash:

  a-z ASCII 0x61 through 0x7A
  0-9 ASCII 0x30 to 0x39
  - ASCII 0x7E

6B.3 Formation of Namespace URIs:

The following rules govern creation of namespace Universal Resource Identifiers (URIs) in PDS; they must:

- meet `xs:anyURI` requirements (see W3C, 2012b)
- be unique across the PDS
- be lower case
- begin with “http://” irrespective of whether the namespace is within the PDS domain
- begin with “http://pds.nasa.gov/pds4/” if the namespace is within the PDS domain
- end with "vN" where N is a version number. In the examples in Table 6B-1 N is 1.

It is important to note that while these namespace URIs look very much like URLs (Universal Resource Locators), as they begin with “http:”, they are not actual addresses of Internet web pages and are not meant to be used in web browsers.

Table 6B-1: Example Namespaces Recognized by PDS

Namespace ID	Uniform Resource Identifier	Steward
Global Namespaces		
xs	http://www.w3.org/2001/XMLSchema XML/v1	World Wide Web Consortium (W3C)
xsi	http://www.w3.org/2001/XMLSchema-instance/v1	World Wide Web Consortium (W3C)
sch	http://purl.oclc.org/dsdl/schematron/v1	Schematron.com
PDS (Common) Namespace		
pds	http://pds.nasa.gov/pds4/pds/v1	PDS Engineering Node
PDS Discipline and Discipline Node Namespaces		

atm	http://pds.nasa.gov/pds4/atm/v1	PDS Atmospheres Node
geo	http://pds.nasa.gov/pds4/geo/v1	PDS Geosciences Node
img	http://pds.nasa.gov/pds4/img/v1	PDS Cartography and Imaging Sciences Node
naif	http://pds.nasa.gov/pds4/naif/v1	Navigation and Ancillary Information Facility
ppi	http://pds.nasa.gov/pds4/ppi/v1	PDS Planetary Plasma Interactions Node
rings	http://pds.nasa.gov/pds4/rings/v1	PDS Ring-Moon Systems Node
rs	http://pds.nasa.gov/pds4/rs/v1	PDS Radio Science Advisor
sbn	http://pds.nasa.gov/pds4/sbn/v1	PDS Small Bodies Node
cart	http://pds.nasa.gov/pds4/cart/v1	PDS Cartography and Imaging Sciences Node
PDS Mission Namespaces		
mpf	http://pds.nasa.gov/pds4/mission/mpf/v1	Mars Pathfinder
PDS Namespaces within Mission Namespaces		
gllnims	http://pds.nasa.gov/pds4/mission/gll/nims/v1	Galileo Near Infrared Mapping Spectrometer
gllssi	http://pds.nasa.gov/pds4/mission/gll/ssi/v1	Galileo Solid State Imaging

6C File and Directory Naming

Although modern operating systems are permissive when it comes to file and directory names, there are good reasons for assigning names conservatively. For example, characters that are legal on one system may need to be 'escaped' on another. Similarly, names with 255 characters, while permitted, are often inconvenient for users who cannot distinguish among similarly constructed, lengthy names in displays and lists. Also, the concatenation of long file and/or directory names may exceed limits for a path name once all components have been combined. Finally, although much data management is carried on by machines, humans will find intuitive names to be much easier to grasp should they need to inspect an archive manually.

In the following sections "prohibited" means "not allowed" while "reserved" means that the name must be employed in a way that is consistent with PDS usage. For example, "geometry" may only be used in a base name (see Section 6C.1.1) or directory name if the content of the file

or directory has an obvious connection to geometry. Case variations of prohibited and reserved names below are similarly constrained.

In this section we use the asterisk "*" to denote a wild card representing zero or more additional characters and square brackets "[]" to enclose options.

6C.1 File Names

6C.1.1 Rules

The following rules govern selection of file names.

- The file name must be unique within a directory. The same file name may appear in different directories.
- The file name must be no longer than 255 characters (and be commensurate with other length limits, such as for path names).
- The file name must be case-insensitive; for example, "MyFile.txt" and "myfile.txt" are not permitted in the same directory.
- The file name must be constructed from the character set

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

- The file name must not begin or end with a dash, underscore, or period.
- The file name must include at least one period followed by an extension. A file name may have more than one period, but PDS will consider all periods other than the final one to be part of the base name.

6C.1.2 Prohibited File Names

The following file names have specific purposes on some operating systems; their use within PDS archives is prohibited.

```
a.out                    core
```

6C.1.3 Reserved File Names

The following file names are reserved for specific purposes within PDS archives and may not be used otherwise:

```
bundle*.[xml,lblx]                    readme*.txt
catalog*.[xml,lblx]                   pds4_pds*.[sch,xsd,xml]
collection*.[xml,lblx,csv]
```

For example, file names starting with "`readme`" and ending with "`.txt`" are reserved while "`readme.abc`" is not.

6C.1.4 Prohibited Base Names

Because the following base names have specific purposes on some operating systems, their use within PDS archives is prohibited.

aux	com5	con	lpt5	nul
com1	com6	lpt1	lpt6	prn
com2	com7	lpt2	lpt7	
com3	com8	lpt3	lpt8	
com4	com9	lpt4	lpt9	

6C.1.5 Reserved Base Name Components

The following terms and abbreviations are reserved for use as components of base names:

browse	geometry	node
calibration	instrument_host	resource
collection	instrument	schema
context	investigation	spice_kernels
data	inventory	target
document	miscellaneous	

6C.1.6 Reserved File Name Extensions

The file name extensions listed in Table 6C-1 are reserved. A brief description for each is provided in the table. The presence of any extension in this table does not imply that the associated format is acceptable in every (or any) collection.

Table 6C-1 – Reserved File Name Extensions

Extension	Description
bc	SPICE binary CK file (spacecraft and instrument orientation data)
bdb	SPICE binary EK file, data base component (event data base data)
bds	SPICE binary DSK file (natural body digital shape data)

bep	SPICE binary EK file, plan (science plan data)
bes	SPICE binary EK file, sequence component (event sequence data)
bpc	SPICE binary PCK file (high-accuracy natural body rotation data)
bsp	SPICE binary SPK file (trajectory and ephemeris data)
csv	delimited tabular data file (comma separated value)
dat	generic binary data file (used for files containing different types of digital data)
doc,docx	Microsoft Word file
fit,fts	Flexible Image Transport System file
gif	Graphics Interchange Format file
hdr	generic header file, usually character data
htm,html	HyperText Markup Language formatted file
img	raster formatted image data (may contain headers)
jp2	JPEG 2000 formatted file
jpg,jpeg	Joint Photographic Experts Group formatted file
lbl	PDS3 or other format label
nrb	SPICE text orbit number file, ascending or descending node numbering (orbit start and stop times)
orb	SPICE text orbit number file, periapsis or apoapsis numbering (orbit start and stop times)
pdf	Portable Document Format
sch	Schematron file
tab	fixed-width tabular data file
ten	SPICE text EK file, notebook component (observation notes data)
tep	SPICE text EK file, plan component (science plan data)
tf	SPICE text FK file (reference frame definitions)
ti	SPICE text IK file (instrument parameters and FOV definitions)
tif,tiff	Tagged Image Format File
tls	SPICE text LSK file (leapsecond information)
tm	SPICE text MK file (meta-kernels listing kernels to be used together)
tpc	SPICE text PCK file (natural body rotation and size/shape constants)
tsc	SPICE text SCLK file (spacecraft clock correlation data)
txt	plain text file

`vic`	Video Image Communication and Retrieval file
`xml,lblx`	Extensible Markup Language formatted file
`xsd`	XML Schema file
`zip`	Zip compressed archive file

6C.2 Directories

6C.2.1 Rules

The following rules govern selection of directory names.

- The directory name must be unique within its parent directory.
- The directory name must be no longer than 255 characters (and be commensurate with other length limits, such as for path names).
- The directory name must be case-insensitive; for example, "MyDirectory" and "mydirectory" are not permitted in the same parent directory.
- The directory name must be constructed from the character set

A-Z	ASCII 0x41 through 0x5A,
a-z	ASCII 0x61 through 0x7A,
0-9	ASCII 0x30 through 0x39,
dash "-"	ASCII 0x2D), and
underscore "_"	ASCII 0x5F

- The directory name must not begin or end with a dash or underscore.

6C.2.2 Reserved Directory Names

The following directory names are reserved for specific purposes within PDS and may not be used elsewhere:

`browse`	`geometry`
`calibration`	`miscellaneous`
`context`	`spice_kernels`
`data`	`xml_schema`
`document`	

6C.2.3 Prohibited Directory Names

Because the following names have specific purposes on some operating systems, their use within PDS archives is prohibited.

`aux`	`com1`	`com2`	`com3`	`com4`

com5	com9	lpt2	lpt6	nul
com6	con	lpt3	lpt7	prn
com7	core	lpt4	lpt8	
com8	lpt1	lpt5	lpt9	

6C.2.4 Directory Path Names

Directory path names are the concatenation of two or more directory names. The directory names are separated with a forward slash '/'. Directory paths in PDS4 labels are always given relative to the directory containing the label and may only point to subdirectories of that directory (i.e., the use of "../" is not permitted).

6D Identifiers

An *identifier* is a character string that is uniquely associated with an object, product, collection, or bundle. Identifiers are strings of no more than 255 ASCII printable characters (see Section 6A.3); the sets of allowed characters depend on the type of identifier.

6D.1 Local Identifier

A *local identifier* is unique within a label. It can be assigned to distinct parts of the label to simplify cross-referencing within the label; for that reason, local identifiers should be easy for humans to read, understand, and trace. The local identifier value must start with either a letter or an underscore. The characters must be from the ASCII alphanumeric character set, underscores, colons, and/or dashes as follows:

upper case letters	A-Z	ASCII 0x41 to 0x5a
lower case letters	a-z	ASCII 0x61 to 0x7a
digits	0-9	ASCII 0x30 to 0x39
dash	"-"	ASCII 0x2D, and
underscore	"_"	ASCII 0x5F

6D.2 Logical Identifier

A *logical identifier* (LID) is uniquely associated with the set of all versions of a bundle, collection, or product. It is constructed as four (bundle), five (collection), or six (product) fields from the following list of ASCII characters:

lower case letters	a-z	ASCII 0x61 to 0x7a
digits	0-9	ASCII 0x30 to 0x39
dash	"-"	ASCII 0x2D,

period	"."	ASCII 0x2E, and
underscore	"_"	ASCII 0x5F

```
urn:nasa:pds:<bundle_id>
urn:nasa:pds:<bundle_id>:<collection_id>
urn:nasa:pds:<bundle_id>:<collection_id>:<product_id>
```

where colons (ASCII 0x3A) delimit the fields (and may not be used within fields), and each field must begin with a letter or digit.

LIDs that identify bundles, collections, or products belonging to agencies other than PDS must begin with the construction

```
urn:<national_agency>:<archiving_agency>:
```

for example,

```
urn:esa:psa:
```

where "esa" indicates the European Space Agency and "psa" indicates "Planetary Science Archive". The Information Model lists the agency tokens permitted for use in LIDs.

6D.3 Versioning

Versioning must be expressed using the format M.n.o.p where M is the most significant field and fields n, o, and p have decreasing significance in left-to-right order. The o and p fields are not used in some applications.

Field values must be incremented as integers; M starts from "1", while the other fields start from "0". Field values must not be prepended with leading zeros. Thus "1.1" and "1.10" are different versions, and "1.01" is invalid. When one field value is incremented, all of the less significant fields must be reset to "0".

6D.3.1 Bundle, Collection, and Product Versioning

Every product submitted to the PDS must have a valid version number. During early stages of product development, the consulting node may agree to permit the provider to submit multiple variations of a product without changing the version number providing these submissions a) are considered preliminary or draft products, b) are not ingested into the PDS4 registry, and c) are not released to the general public. Every product (basic or aggregate) submitted to the PDS for ingestion into the PDS4 registry, and every product made available to the user community must have a unique LIDVID, and that LIDVID may not be the same as a previously submitted version unless the product is identical to the prior submission. Even minor changes to a product - label or data object - require incrementing the version identifier.

A *version identifier* (VID) of the form `M.n` may be appended to a logical identifier to select one of several versions of the same bundle, collection, or product. The combination is called a

versioned identifier (LIDVID). LIDVIDs are used to locate products within PDS; every version of every product within PDS has a unique LIDVID.

VIDs are separated from LIDs by a double colon ("::") and have the form `M.n` where `M` and `n` are both integers.

```
urn:nasa:pds:<bundle_id>::<version_id>
urn:nasa:pds:<bundle_id>:<collection_id>::<version_id>
urn:nasa:pds:<bundle_id>:<collection_id>:<product_id>::<version_
id>
```

Bundle, collection, and product version identifiers are issued sequentially, although not all version numbers may appear in an archive. `M` denotes a major version and `n` denotes a minor version. `M` is initialized to "1" for archives, but "0" may be used for samples and tests; `n` is initialized to "0".

The addition or deletion of any product (primary or secondary) to a collection requires an increment in the collection VID because the collection Inventory has changed. If an existing product in the collection is modified, the collection version is incremented if the Inventory lists the product by LIDVID, and is not incremented if the product is listed by LID. The same is true for a collection within a bundle; the bundle VID must be incremented if any of its collections are added or deleted; or if any of its collections identified by LIDVID are modified.

6D.3.2 System, Namespace, Data Dictionary, and Document Versioning

Although versioning is limited primarily to bundles, collections, and products in the Information Model, it is also used for the major components of the PDS4 system itself and for documentation.

For the PDS4 common dictionary, the `M.n.o.p` format must be used, where the field significance follows conventions for XML documents:

- `M` is incremented when one or more requirements is added and/or changed.
- `n` is incremented when one or more options is added and/or changed. A requirement which is relaxed to an option triggers an increment in `n` rather than `M` since the result should be backwardly compatible.
- `o` is incremented when one or more enumerated values is added and/or changed.
- `p` is incremented when text unrelated to requirements, options, or enumerated values is added, removed, and/or changed. When `o` is used and is incremented, `p` is reset to '0'; when `o` is not used and `n` is incremented, `p` is reset to '0'.

Versions `1.n.o.p` of the system, namespaces, data dictionaries, and documents were developed before the above rules were promulgated and do not adhere rigorously to the above requirements.

Versioning of discipline and mission level dictionaries is more flexible. Developers of these dictionaries may, if they wish, use the same criteria as described above for the common dictionary. Alternatively, some of the components of the version may be used to synchronize the dictionary with a higher level dictionary on which it has dependencies. Discipline and mission level dictionary versions may have anywhere from one (`M`) to four (`M.n.o.p`) components.

The common part of the PDS4 Information Model and the associated local data dictionaries (and all of their respective documentation) may each have their own versioning sequences. Aggregations of these components (the 'total' PDS system) are versioned separately.

Documents are updated and released as necessary. Versioning of those documents will be coarsely synchronized with the versions of the system components they describe

6E Classes, Attributes, and Attribute Values

6E.1 Classes

The following rules govern the selection of class names throughout PDS.

- Class names must be constructed from one or more components, each of which is made from members of the ASCII alphanumeric character set (see Section 6A.2).
- Class names must clearly convey meaning and pertinence (*e.g.*, Record_Binary).
- Components must be separated by ASCII underscores.
- Each component must begin with an upper case letter; all other letters must be lower case (*e.g.*, Product_Collection) except when the class name incorporates an acronym (*e.g.*, SPICE_Kernel).
- The most significant component must be placed first with subsequent components in decreasing order of importance (*e.g.*, Product_Table_Character rather than Character_Table_Product).
- Articles and prepositions must not be used as components.
- Components must clearly convey meaning and pertinence (*e.g.*, Record_Binary).
- When practical, subclass names must include part of the parent class name (*e.g.*, Array_2D_Image is a subclass of Array_2D, which is a subclass of Array).
- The class name must not exceed 255 characters.

6E.2 Attributes

The following rules govern selection of attribute names throughout PDS.

- Attribute names must be constructed from one or more components, each of which is made from lower-case ASCII letters and/or digits
- Attribute names must begin with a lower-case ASCII letter.
- Attribute names must clearly convey meaning and pertinence (*e.g.*, institution_name).
- Components must be separated by ASCII underscores.
- The least significant component must be placed first with subsequent components in increasing order of importance (*e.g.*, maximum_value).
- Articles and prepositions must not be used as components

- Components must clearly convey meaning and pertinence (*e.g.*, institution_name).
- The attribute name must not exceed 255 characters.

6E.3 Attribute Values

Attribute values should be lower case with capitalization consistent with normal usage — that is, lower case except for acronyms, abbreviations, IDs, the first character in proper names, the first character at the beginning of a sentence, etc. Single words or short phrases should generally be lower case. Underscores should not be used as separators unless they would appear in usage outside labels. However, PDS will not necessarily reject attribute values that fail to meet these guidelines; the guidelines are primarily to help humans read label information.

PDS searches on attribute values will be case sensitive for enumerated values and case insensitive otherwise.

6E.3.1 Enumerated Attribute Values

Enumerated values must be given in 'title case' except when the value is an acronym, abbreviation, or modeling term or has been carried over from PDS3. In title case the first letter of each word (except for articles, prepositions, and conjunctions) is capitalized. Acronyms and abbreviations are capitalized as in normal usage (*e.g.*, NASA, MESSENGER, HiRISE, and InSight). Modeling terms are imported directly from the PDS4 Information Model or a NAIF model and are usually distinguished by underscores linking words or identifiers (*e.g.*, attribute_of, bundle_has_browse_collection, Units_of_Temperature, and MOON_ME_DE421).

Case variations of the same character string may not be used as enumerated values of an attribute. That is, "Single" and "SINGLE" must not be enumerated values of the same class.attribute pair even if their meanings are different. Hexadecimal numbers must be all lower case or all upper case, but not both.

6E.3.2 Attribute Value Units

Attributes that are measurable quantities — *e.g.*, <altitude>, <file_size>, and <latitude> — must have the unit of measurement specified by an XML attribute 'unit' in the tag. The unit must be selected from the set of possible units given by <unit_id> in the appropriate Unit_Of_Measure subclass. For example:

```
<altitude unit="m">173.293</altitude>
```

where the possible choices from Units_of _Length are:

Angstrom nm micrometer mm cm m km AU

6E.3.3 Attribute Value Limits

Attributes may be limited to a specific set of values if specified in the defining data dictionary. Specific values are set (and their meanings defined) by repeated use of the DD_Permissible_Value class in the data dictionary definition. For example, the attribute <encoding_type> may have the values `binary` or `character` depending on the class in which it is used.

Limits on the range of attribute values (and/or maximum or minimum numbers of characters in character strings) may be imposed by using the meta-attributes <maximum_value> and <minimum_value> (or <maximum_characters> and <minimum_characters>) in the DD_Value_Domain class. Units for permissible values, <maximum_value>, and <minimum_value> are set by meta-attribute <specified_unit_id> in the data dictionary, where the <specified_unit_id> must be selected from the possible values given by <unit_id>.

7 Units

Uniform use of units facilitates catalog searches across archive systems. The PDS standard for units is the *Systeme Internationale d'Unites* (SI); see http://physics.nist.gov/cuu/Units/units.html for details.

There are two levels at which units are specified — the type of measurement (*e.g.*, length, mass, time, etc.) and the specific units in which a measurement is reported (*e.g.*, meters, kilograms, seconds, etc.) — and three applications (data dictionaries, attribute values in labels, and data objects).

7A Types of Measurement

When an attribute denoting a measurable quantity is added to a PDS data dictionary, the type of measurement must be specified using the <unit_of_measure_type> meta-attribute. Valid choices (among many others) include the classes

Units_Of _Length

Units_Of _Mass

Units_Of _Time

The definition of each class includes the attributes <type> (the last field in the class name—*e.g.*, length, mass, time, etc.), <unit_id> (the set of acceptable abbreviated units for use in labels), and <specified_unit_id> (the unit in which meta-attributes <maximum_value>, <minimum_value>, and <value> are given within the `DD_Value_Domain_Full` specification). Examples are shown in the table below:

type	unit_id
acceleration	cm/s**2 km/s**2 m/s**2
angle	Arcmin arcsec deg hr mrad rad
length	AU Angstrom cm km m micrometer mm nm

7B Specific Units

For attributes denoting measurable quantities, the specific unit must be selected in the label using an XML attribute; the XML attribute <unit> identifies the selected unit from the possibilities provided by <unit_id> in the data dictionary.

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

In the XML statement above, <length> is the attribute, "unit" is the qualifying XML attribute, and "km" is its value — the specific unit. That is, the value of <length> is 3 km.

7C Prefixes

Prefixes (or their abbreviations) shown in the table below may be prepended to any SI unit.

SI PREFIXES

Factor	Prefix	Symbol	Factor	Prefix	Symbol
10**24	yotta	Y	10**-1	deci	d
10**21	zetta	Z	10**-2	centi	c
10**18	exa	E	10**-3	milli	m
10**15	peta	P	10**-6	micro	
10**12	tera	T	10**-9	nano	n
10**9	giga	G	10**-12	pico	p
10**6	mega	M	10**-15	femto	f
10**3	kilo	k	10**-18	atto	a
10**2	hecto	h	10**-21	zepto	z
10**1	deka	da	10**-24	yocto	y

There is presently no ASCII symbol for “micro”.

7D Derived Units

If the needed specific unit is not available from the PDS data dictionary, PDS allows construction of a new unit by concatenating units and prefixes that have been approved and common mathematical symbols (“-“ for negation, “**” for exponentiation, “*” for multiplication, and “/” for division in that order of precedence). In this way a data provider can express flux in watts per square meter (W/m**2), for example. Note that map scales and radiance, examples of other parameters using compound units, are already in the PDS data dictionary.

7E Expressions

Units are written in lower case except for those derived from proper names or acronyms or which have upper case symbol prefixes. No periods are used in any of the abbreviations.

Expressions should be written compactly and unambiguously with (at most) one pair of parentheses enclosing factors in the numerator, one pair of parentheses enclosing factors in the denominator, a division operator "/", and no negative exponents — e.g.," m/s", "m/s**2", or "(kg*m**2)/(s**2)". Prefixes must be reduced — *i.e.*, although “mkm” and “m” are numerically identical, the first is not allowed.

8 Content

PDS requires that products, collections, and bundles be documented so that scientists in future years can understand (1) how the data were collected and processed, (2) what the data represent, and (3) the limitations of the data. These future scientists are assumed to have backgrounds comparable to the people who archived the data; they should be able to assess the value of the data in new work and then use the data. This section summarizes *what* should be included in PDS archives as opposed to how it is presented, which was the focus of Sections 1-7; the emphasis is on documentation.

8A Documentation Requirements

Documentation takes three forms in PDS: (a) XML labels, (b) documents included within the archive, and (c) references to material that is publicly available elsewhere.

Data providers must negotiate with the PDS consulting node early in the archiving process on the documentation requirements and how they will be distributed among the three categories above. Peer review provides feedback on whether the goals have been achieved. Questions to be asked at each step include:

- What is necessary for evaluation, understanding, and use of the data?
- What might be useful beyond the 'necessity' threshold?
- What is publicly available?
- How should documentation be divided among labels, internal documents, and external documents?

The contents of the documentation must include (but are not limited to) detailed descriptions of the instrument, spacecraft or other instrument 'host', investigation(s), data acquisition procedures, and analysis procedures, facilities, and personnel. If additional processing is expected to be performed by either the data provider or the user, those steps should be outlined.

8A.1 Labels

Labels provide machine-readable information about structure and relationships. The label structure is discussed in more detail elsewhere (see Section 3).

Labels also provide references. This is the primary mechanism by which the relationships are established among products within and external to the archive. An Internal Reference is used to establish references to products (*e.g.*, document products, context products, other collection products, etc.) that are within the PDS repository. An External Reference is used to establish references to products that are available external to the PDS (*e.g.*, material that is publicly available elsewhere).

8A.2 Internal Documentation

Internal documentation is one or more document products that have been included in a bundle. As with other products, each document product must be a primary member of one collection (and may be a secondary member of many). Most document products will be members of a document

collection, which is then a member of a bundle; however, some documents may appear elsewhere, such as in a calibration or miscellaneous collection. Bundles are not required to have document collections, but it would be unusual for a bundle not to include at least one document collection.

8A.2.1 Document Collections

Data collections may be formed in a variety of ways. For most data submissions, a single Document Collection, possibly organized with multiple subdirectories, will be sufficient. In other cases, it may be preferable to separate documents of different types into a few collections. For especially complex documents, it may be most practical to assign all the pieces (text, figures, tables, etc.) to a single document collection — in which case the result may be one document per collection. In other cases, it may be sufficient to separate documents of different types into a few collections.

Document products related to calibration may be primary members of a calibration collection rather than a document collection. Documents related to 'geometry' may be primary members of a geometry or SPICE kernels collection rather than a document collection.

8A.2.2 Document Products

The range of relevant documents will depend on the type(s) of data and the scope of the associated collections and bundle. In general, completeness of information is the goal. For a typical planetary spacecraft mission, documents could include:

- Mission Plan
- Instrument papers
- Instrument user guides
- Calibration reports
- Science articles
- Archive content, format, and organization, e.g. Software Interface Specifications (SISs)
- Software user manuals

Including a document in a PDS archive constitutes publication (or re-publication) of that document. Consequently, documents must meet not only the PDS label and format requirements, but also the structural, grammatical and lexical requirements of a refereed journal.

It is the responsibility of the data provider to obtain clearance for materials judged to be sensitive under U.S. International Traffic in Arms Regulations (ITAR) and to obtain permission from copyright holders for re-use of protected materials in the archive. PDS reserves the right to request documentation of either or both ITAR and copyright permissions.

Each document must be saved in one or more PDS-compliant formats using the Product_Document class. PDS-compliant formats for documents are "flat UTF-8" text, PDF/A-1a (which is preferred), or PDF/A-1b. Figures may be embedded in PDF files. However. any figures other than those embedded in a PDF file must be formatted as JPEG, GIF, PNG or TIFF images. "Flat UTF-8" means the file may not contain mark-up language (*e.g.*, formats such as HTML, XML, RTF, and LaTeX do not comply).

For documents, there can be only one data object per file.

Video format files may be included as document products (and only as document products) provided that they are supplemental material and are not intended to replace textual documentation. Videos are subject to the following restrictions:

- The file format must be MPEG-4 Part 14 (.mp4)
- The video codec must be H.264
- The audio codec must be AAC.

Tools for both converting videos to this format and validating that videos conform to these restrictions are freely available.

8A.3 External Documentation

For archive integrity, it is preferable to have all relevant documentation within the archive; in many cases, however, this is either impractical or impossible. For example, when a copyright holder refuses re-use, the copyrighted material must be referenced through an outside source.

The Reference_List class may be used to point toward relevant material or source products either inside or outside PDS. It may be associated with any of over two dozen different product classes including Product_Context, Product_Document, Product_Collection, and Product_Browse.

The External_Reference class provides a complete bibliographic citation to a published work outside the archive, optionally including its Digital Object Identifier (DOI). External_Reference may be used through Reference_List or directly, such as from Observing_System_Component. External_References to inaccessible materials may lead to rejection during peer review.

The Source_Product_Internal class is used to reference source products in the PDS4 registry system. Source_Product_External is used to reference source products not in the PDS4 registry system, including legacy PDS3 products and products from non-PDS archives, such as the NSSDCA Master Catalog holdings. In cases where a label needs to reference many source products, the classes Table_Delimited_Source_Product_Internal and Table_Delimited_Source_Product_External may be used to refer to a table containing the source product information.

Records in these tables are delimited either by the line-feed character or by carriage-return line-feed pairs. Records in a Table_Delimited_Source_Product_Internal each have two fields, while records in a Table_Delimited_Source_Product_External each have four fields. Fields are delimited by a single comma.

For both Table_Delimited_Source_Product_Internal and Table_Delimited_Source_Product_External, the first field contains the reference type of the source product. The reference type must be one of the following:

```
'data_to_telemetry_source_product',
'data_to_raw_source_product',
'data_to_partially_processed_source_product',
'data_to_calibrated_source_product',
'data_to_derived_source_product'.
```

In a Table_Delimited_Source_Product_Internal, the second field contains the LIDVID of the source product.

In a Table_Delimited_Source_Product_External, the second field contains the product identifier of the external source product, the third field contains the DOI of the source product, and the fourth field contains the curating facility of the source product. Each record in a Table_Delimited_Source_Product_External must contain at least one of DOI or curating facility.

A table of internal source products might look like this:

```
data_to_calibrated_source_product, urn:nasa:pds:example:calibrated:example_cal_1::2.0
data_to_calibrated_source_product, urn:nasa:pds:example:calibrated:example_cal_2::1.0
data_to_calibrated_source_product, urn:nasa:pds:example:calibrated:example_cal_3::1.0
data_to_derived_source_product, urn:nasa:pds:example:derived:example_derived_1::1.0
data_to_derived_source_product, urn:nasa:pds:example:derived:example_derived_2::1.0
```

And a table of external source products might look like this:

```
data_to_derived_source_product, GO-J/JSA-SSI-2-REDR-V1.0:go_0017/c3/callisto/c036831/:7400r.img,, JAXA
data_to_derived_source_product, GO-J/JSA-SSI-2-REDR-V1.0:go_0017/c3/callisto/c036831/:7500r.img, 13.4321/2016JE987654322, JAXA
```

Note that the first record in the external table example does not contain a DOI, and thus there are two commas before the curating facility.

8B Context Bundle, Collections, and Products

"Context" products are terse summaries of (and links to) other documentation that can help users navigate within PDS; they contain nothing that is essential to understanding and using PDS data that cannot be found elsewhere. However, they can represent a simple source for associating references to key physical and conceptual objects. Consequently, while most context products are prepared by data providers, they become members of collections for which the PDS Engineering Node is the steward. Those collections are members of a single PDS 'master' context bundle which is used widely across the system. Context products in a namespace belonging to an international agency are controlled by the steward of that agency.

8B.1 Collections Under the PDS4 Context Bundle

The PDS 'master' context bundle contains separate 'master' context collections, one for each type of context product (*e.g.*, instrument, telescope, facility, PDS Node, etc.).

8B.2 PDS Context Products

A physical or conceptual object referenced by a PDS product must have its own context product; for example, every spacecraft will have a context product and each instrument on the spacecraft will have a context product. Every context product must be a primary member of one of the collections in the master context bundle.

A data provider — working with a consulting node — is generally responsible for providing the content for a context product. However, since the PDS Engineering Node is the steward for the entire master context bundle, the provider must also work closely with EN personnel on product construction, starting with LID and VID assignments for each new context product.

A context product is a single XML label for the physical or conceptual object. The contents of the label are simple XML statements that give the versioned identifier (LIDVID) of the product, the name and abbreviation (if any) of the object, the type of object (*e.g.*, instrument, node, target, etc.), a brief description of the object suitable for display within a web interface, and reference

associations for the critical documents describing the object.

8B.3 Context Collections and Products in Science Bundles

A data provider may optionally include a context collection when submitting a bundle containing scientific data to PDS. That context collection may be either a primary or secondary member of the bundle. The collection's own membership is specified in its Product_Collection – a collection inventory table and its associated XML label.

The context products associated with the collection — even those shepherded to completion by the data provider — must be secondary members of the science bundle's context collection. The submission details (delivery method and timing) for new context products are negotiated among the provider, the consulting node, and EN. As noted in Section 2A.4, secondary members of a context collection need not be included in deliveries to the PDS.

When science bundles do not include context collections, references must be to context products and collections in the PDS4 Context Bundle.

8C Calibration

Instruments are imperfect sensors; their outputs require adjustment so that the measurements are quantitatively meaningful, presented in physical units, and as accurate as possible. The details will vary among instruments types and disciplines; data providers should negotiate archiving requirements with consulting nodes. Calibration products would ordinarily be members of a calibration collection (see Section 2A.5), but other options are possible — such as when calibration data are interleaved with science observations.

8D Geometry

Planetary flight missions are expected to archive complete geometric details from launch through end of mission. These typically include full ephemerides of the spacecraft and relevant planetary bodies, orientation of the spacecraft and all instruments, the relationship of these to coordinate systems on the target(s), a history of all significant spacecraft events, and other 'housekeeping' data (such as temperatures and power levels) that might be useful in understanding the behavior of instruments. Radiometric tracking data should be archived even if there is no 'instrument team' for radio science.

The archives of Earth-based observing campaigns and other observations must include equivalent geometrical data. The objective in each case is to allow future data users to reconstruct the observing geometry in a way that supports quantitative analysis.

Geometry products should be members of a geometry collection or a SPICE kernels collection; the distinction is described in Section 2A.5.

9 Special Constraints

9A Header Class

All information necessary to read, understand, and work with data in a PDS product must be available in the product label without reference to an accompanying Header object. The Header class, which is optional, is provided for users working in non-PDS environments.

9B Group Fields and Groups

The attributes <fields> and <groups> in classes Group and Record may each have a minimum value of "0", but their sum must be at least "1".

9C Product_Collection

Collections are defined by Product_Collection — a character table with delimited records, called an Inventory table, and an accompanying label. Each record in the table identifies a basic product in the collection by member status (primary or secondary) and LID or LIDVID. For every member in the collection there must be one and only one record in the table.

Any single version of a collection is defined and uniquely identified using the Product_Collection class definition — an XML label file paired with an inventory table. Collection members must use either a ".xml" or ".lblx" extension for the XML label files; but not a mix within a single Collection.

9C.1 Creating an Inventory Table

Records in the Inventory table are delimited either by the line-feed character or by carriage-return line-feed pairs; each record has two fields, delimited by a single comma.

The first field contains the member status, denoted by a single character. 'P' indicates that the product is a primary member of the collection; 'S' indicates that the product is a secondary member. The second field contains the LID or LIDVID for a basic product.

The example below shows six records from a hypothetical collection inventory. The first three rows are for primary members; primary members must be identified using a LIDVID. The next two are for secondary members, each identified using a LID. The last entry is a secondary member, identified using a LIDVID.

```
P,urn:nasa:pds:dph:data-cal:prod1::1.1
P,urn:nasa:pds:dph:data-cal:prod2::1.4
P,urn:nasa:pds:dph:data-cal:prod3::1.1
S,urn:nasa:pds:dph:data-raw:prod1
S,urn:nasa:pds:dph:data-raw:prod2
S,urn:nasa:pds:dph:data-raw:prod3::1.2
```

9C.2 Generating and Populating a Product_Collection Label

The collection inventory is defined by an Inventory object description in the label, which itself has a Record_Delimited and two Field_Delimited object descriptions. There are additional constraints on attribute values beyond those specified in the Information Model. In the example below, lines shown in bold are required exactly as shown. Unbolded lines are required, but values may vary. Optional lines are not shown.

Values for offset and for records should be the appropriate integers for the inventory table being described. The value for field_delimiter must be "comma". The value for name in the first field definition must be "Member Status"; the value for name in the second field definition must be one of "LID", LIDVID", or "LIDVID_LID", matching the value of data_type less "ASCII_".

```
<Inventory>
   <offset unit="bytes">0</offset>
   <parsing_standard_id>PDS DSV 1</parsing_standard_id>
   <records>10</records>
   <record_delimiter>Carriage-Return Line-Feed</record_delimiter>
   <field_delimiter>Comma</field_delimiter>
   <Record_Delimited>
      <fields>2</fields>
      <groups>0</groups>
      <Field_Delimited>
         <name>Member Status</name>
         <field_number>1</field_number>
         <data_type>ASCII_String</data_type>
         <maximum_field_length unit="byte">1</maximum_field_length>
      </Field_Delimited>
      <Field_Delimited>
         <name>LIDVID_LID</name>
         <field_number>2</field_number>
         <data_type>ASCII_LIDVID_LID</data_type>
         <maximum_field_length
         unit="byte">255</maximum_field_length>
      </Field_Delimited>
   </Record_Delimited>
   <reference_type>inventory_has_member_product</reference_type>
</Inventory>
```

In addition, the Citation_Information class in the Identification_Area is required. The description attribute in Citation_Information provides a terse description of the contents of the collection suitable for display in a web browser.

When a collection has a Digital Object Identifier (DOI) assigned, indicated by the presence of the doi attribute in the Citation_Information, the author_list (or_editor_list) attribute of Citation_Information must be included.

9C.3 Constraints on Collections

Constraints have been placed on certain types of collections to ensure that the metadata needed for search and discovery are present in the XML label. The constraints are documented in this section.

9C.3.1 Mission Science Data Collection - Constraints

If the following metadata values are present, this is a Mission Science Data Collection:

- Product_Collection/Collection/collection_type: the value is ‘Data’
- Product_Collection/Context_Area/Investigation_Area/type: at least one has value 'Mission'
- Product_Collection/Context_Area/Primary_Result_Summary/purpose: at least one has value 'Science'

Then the following must be present; in some cases multiple instances of the attribute may appear in the label.

- Product_Collection/Identification_Area/Citation_Information/description
- Product_Collection/Context_Area/Observing_System/Observing_System_Component/type: at least one must have value 'Instrument'
- Product_Collection/Context_Area/Time_Coordinates/start_date_time
- Product_Collection/Context_Area/Time_Coordinates/stop_date_time
- Product_Collection/Context_Area/Target_Identification/name
- Product_Collection/Context_Area/Target_Identification/type
- Product_Collection/Context_Area/Investigation_Area/name
- Product_Collection/Context_Area/Observing_System/Observing_System_Component/name
- Product_Collection/Context_Area/Primary_Result_Summary/processing_level

9D Product_Bundle

Bundles are defined by Product_Bundle, which contains an XML label that defines the bundle membership and (optionally) a "readme" file that gives an overview of the bundle. For every member in the bundle there must be one and only one Bundle_Member_Entry in the label; member collections are identified by LID or LIDVID, as appropriate.

9D.1 Creating a Product_Bundle

If Product_Bundle includes an optional "readme" file, the optional file must be a parsable byte stream. An ASCII or UTF-8 text file that describes the bundle, its scope, and its processing history would qualify.

9D.2 Generating and Populating a Product_Bundle Label

The Product_Bundle label must include one and only one Bundle_Member_Entry for each member collection. Either a lid_reference or a lidvid_reference attribute (but not both) must be included in each Bundle_Member_Entry definition.

The Citation_Information class in the Identification_Area is required. The description attribute in Citation_Information provides a terse description of the contents of the bundle suitable for display in a web browser.

When a bundle has a Digital Object Identifier (DOI) assigned, indicated by the presence of the doi attribute in the Citation_Information, the author_list (or_editor_list) attribute of Citation_Information must be included.

9D.3 Constraints on Bundles

Constraints have been placed on certain types of bundles to ensure that the metadata needed for search and discovery are present in the XML label. The constraints are documented in this section.

9D.3.1 Mission Science Archive Bundle - Constraints

If the following metadata values are present, this is a Mission Science Archive Bundle:

- Product_Bundle/Bundle/bundle_type: the value is ‘Archive’.
- Product_Bundle/Context_Area/Investigation_Area/type: at least one has value 'Mission'.
- Product_Bundle/Context_Area/Primary_Result_Summary/purpose: at least one has value 'Science'.

Then the following must be present; in some cases multiple instances of the attribute may appear in the label.

- Product_Bundle/Identification_Area/Citation_Information/description
- Product_Bundle/Context_Area/Time_Coordinates/start_date_time
- Product_Bundle/Context_Area/Time_Coordinates/stop_date_time
- Product_Bundle/Context_Area/Target_Identification/name
- Product_Bundle/Context_Area/Target_Identification/type
- Product_Bundle/Context_Area/Investigation_Area/name
- Product_Bundle/Context_Area/Observing_System/Observing_System_Component/name
- Product_Bundle/Context_Area/Observing_System/Observing_System_Component/type: at least one must have value 'Instrument'
- Product_Bundle/Context_Area/Primary_Result_Summary/processing_level

9E Product_Native

Product_Native will be used only for observational data which (1) are in the original format returned by an experimental system or spacecraft and (2) cannot, in this format, be described by Product_Observational. These data are considered to be optional and supplementary; they do not satisfy any obligations the data provider has to archive with PDS.

PDS or any discipline node may reject Product_Native submissions for any reason, including a general local policy of not wanting to curate them.

In Product_Native the file_area association is required; it may occur more than once and must, in each instance, have the value File_Area_Native. In File_Area_Native the association has_tagged_data_object is required; it may occur more than once and must, in each instance, have the value Encoded_Native.

Within the above structure Encoded_Native, a subclass of Encoded_Byte_Stream, includes the required attribute encoding_standard_id which is limited to a list of permissible values. Each permissible value must be approved explicitly by the PDS Management Council. An approved permissible value definition must include identifier(s) for documentation which will allow a user to understand the structure and meaning of the data object contents at the bit level.

Additional information about structure and content must be included in Product_Native.Reference_List including LIDs and/or LIDVIDs to the documentation referenced in the encoding_standard_id.

For each Product_Native a Product_Native.Reference_List.Internal_Reference is required that references an associated Product_Observational using a LID or LIDVID and that has a reference_type value of "native_to_archival". The associated Product_Observational must have scientifically equivalent content.

PDS will verify the accuracy of the encoding_standard_id and Reference_List information at the time of peer review but is not obligated to confirm the accuracy of the documentation itself.

PDS is obligated only to store and maintain (*e.g.*, periodic checksum validations) Product_Native submissions and to support Search and Retrieval services. Search may be limited by the granularity of label information; these holdings are not required to be on-line, so Retrieval may be slow. PDS is not obligated to provide conversion tools, algorithms, or consultation to users interested in working with Product_Native submissions.

The tracking attribute <archive_status> will be defined for all PDS4 holdings; its values will be the same as for its use within the PDS3 migration context. For Product_Native, the only allowed values are IN_QUEUE, IN_QUEUE_ACCUMULATING, SAFED, and SUPERSEDED.

No Product_Native will ever be given the status "certified" under the PDS policy adopted on 2010-09-21, as amended.

9F Class Selection

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

This requirement does not apply to composite structures that are defined using a combination of the class <Local_Internal_Reference> and Local Data Dictionaries (LDDs). Any appropriate subclass can be used.

9G Product_Metadata_Supplemental

Product_Metadata_Supplemental is used to provide additional, or improved, metadata for products in a collection. There is no implicit or explicit intention to generate updated, replacement labels for the basic products affected by a Product_Metadata_Supplemental.

Product_Metadata_Supplemental consists of a file containing a single table (either Table_Character or Table_Delimited), and its XML label. The file containing the table may also contain a Header object that provides the field names for the table.

When a Product_Metadata_Supplemental is produced exterior to the data collection which it supports, it should (recommendation, not requirement) be included as a secondary member of the supported data collection, or the collection of which it is a primary member should be added as a secondary member of the bundle of which the data collection is a primary member.

All field names given in Product_Metadata_Supplemental must be attributes defined in PDS4, either in the common dictionary or in a PDS4 discipline or mission dictionary. For field names that are attributes defined outside the common dictionary, the PDS approved namespace abbreviation of the dictionary in which the attribute is defined must be included in the field name in the Product_Metadata_Supplemental label; e.g., <Field_Delimited> … <name>ladee:delta_time</name>.

The table in Product_Metadata_Supplemental, whether character or delimited, is constructed with one or more records per targeted basic product.

- The first field must be the LIDVID of the targeted basic product.
- If there are multiple records for a single product, the second and if necessary additional consecutive field(s) must be used to distinguish between entries for the same product.

For example, in a collection of long (multi-hour) exposures, Product_Metadata_Supplemental might be used to provide sets of geometric attribute values at one hour intervals during the observation. In the Product_Metadata_Supplemental table, the first field must be the LIDVID of the basic product, the second field must be the time for which the values were calculated, and the following fields provide the individual geometric attributes. A product with a three hour exposure time would have four records in the table, with the value for the time field incremented one hour in each subsequent record:

- urn:…::*1, 2016-02-14T12:20:05.5, …
- urn:…::*1, 2016-02-14T13:20:05.5, …
- urn:…::*1, 2016-02-14T14:20:05.5, …
- urn:…::*1, 2016-02-14T15:20:05.5, …

Product_Metadata_Supplemental LIDs will be of the form: urn:nasa:pds:<bundle>:<collection>:metadata_*.

Versioning: Product_Metadata_Supplemental submissions which include attributes provided in a previous Product_Metadata_Supplemental will be new versions of that previous Product_Metadata_Supplemental.

Submission of a Product_Metadata_Supplemental is not restricted to the initial data provider. Other possible submitters include PDS nodes and successful R&A proposers. A Product_Metadata_Supplemental must be submitted through the PDS discipline node responsible for maintaining the associated collection.

9H Composite Structure

The Composite_Structure class specifies relationships between pairs of digital objects (components) in a single file of observational data using their local identifiers in the corresponding label. The Composite_Structure class additionally may specify relationships between components and description objects in the corresponding label. One, and only one, component must be declared the primary component. Each component must have at least one explicitly declared relationship with another component. There is no requirement that every digital object in the file be a component.

Composite_Structure may appear only within File_Area_Observational or within File_Area_Observational_Supplemental. Components are specified using Local_ID_Reference; relationships are specified using Local_ID_Relation.

10 General Policies

Policies that have been adopted by the PDS Management Council and apply to PDS4 data submissions and review may be found at

https://pds.nasa.gov/datastandards/documents/policy/

Change Log

Version 1.0.0

Date	Section(s)	Change(s)
2013-05-01	All	Original release (Version 1.0.0)

Version 1.2.0

Date	Section(s)	Change(s)
2014-03-20	Change Log	Added Change Log
2014-03-20	6E.3	CCB-22: edited first sentence to be more specific.
2014-03-20	6E.3.1	CCB-22: added new subsection “Enumerated Attribute Values”; renumbered following subsection to 6E.3.2
2014-03-20	4A.5	CCB-29: Replaced last sentence in second paragraph, removing reference to deprecated Display_2D_Image
2014-03-20	5C.4	CCB-34: Third paragraph, second sentence – corrected “negative” to “1”.
2014-03-20	9	CCB-35: Added new chapter “Special Constraints”
2014-03-20	9A	CCB-35: Added new section “Header Class”
2014-03-20	6A.4	CCB-37: Added new subsection defining UTF-8
2014-03-20	Table 5B-1	CCB-37: Changed “UTF” to “UTF-8” in lower right cell
2014-03-20	Table 5C-4 (now Table 5A-4)	CCB-37: Changed “UTF-8” to “UTF-8 string” in lower right cell
2014-03-20	2A.1 and 2A.2	CCB-38: Edited third paragraph in 2A.1 and fourth sentence in 2A.2 to improve clarity.
2014-03-20	7D	CCB-40: Changed “w/m**2” to “W/m**2”
2014-03-20	6D.3	CCB-48: Expanded and reorganized subsection to cover versioning beyond bundles, collections, and products. The first two paragraphs and all of 6D.3.2 are new. 6D.3.1 contains most of the old text with minor edits.

Date	Section(s)	Change(s)
2014-03-20	9C and 9D	CCB-49: Added Sections 9C and 9D, which constrain construction of inventory tables and labels for collections and bundles.
2014-03-20	2A.1	CCB-49: Third paragraph – changed reference for more information from Data Provider's Handbook to Section 9D
2014-03-20	2A.2	CCB-49: changed reference for more information from Data Provider's Handbook to Section 9D
2014-03-20	2B.2.2.2	CCB-49: first paragraph — Changed reference for more information about constructing collection tables from Data Provider's Handbook to Section 9C.1
2014-03-20	1.5	CCB-52: Corrected XML 1.1 2nd edition to XML 1.0 5th edition
2014-03-20	1.5	CCB-52: Removed XML Schema Part 0
2014-03-20	1.5	CCB-52: Updated XML Schema Part 1 from 2004b to 2012a
2014-03-20	1.5	CCB-52: Updated XML Schema Part 2 from 2004c to 2012b
2014-03-20	5A.1, 5A.3, 5A.4, Table 5C-4, Table 5B-1	CCB-52: Updated (W3C, 2004c) to (W3C, 2012b)
2014-03-20	1.5	CCB-52: Added to the ISO section "ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Languages (DSDL) -- Part 3: Rule-based validation -- Schematron"
2014-03-21	1	CCB-47, row 2: Paragraph 1 – dropped single quotes around 'migrated'
2014-03-21	1.1	CCB-47, rows 3 and 6: dropped single quotes around 'common'
2014-03-21	1.1	CCB-47, row 4: Sentence 3 – changed "nodes (DNs)" to "node (DN)"
2014-03-21	1.1	CCB-47, row 5: changed parentheses to long dashes
2014-03-21	1.4	CCB-47, row 7: Sentence 1 – changed "main sections" to "parts"
2014-03-21	1.6	CCB-47, row 8: Underlined first URL for consistency with others.
2014-03-21	2A.1	CCB-47, row 9: Added "(their LIDs/LIDVIDs must be distinct)" to second paragraph.

Date	Section(s)	Change(s)
2014-03-21	2B.1	CCB-47, row 10: Added footnote after “elsewhere” to the PDS4 system specification.
2014-03-21	2B.1.1	CCB-47, row 11: Added “physical” to first sentence
2014-03-21	2B.2.2.2	CCB-47, row 12: Replaced “vid.” by “see” in second paragraph.
2014-03-21	3	CCB-47, row 13: Reworded the last two sentences in the second paragraph.
2014-03-21	3	CCB-47, row 14: Changed first sentence in next to last paragraph from “Figure 3-1 shows the general structure of a label, simplified to components that appear in many labels.” to read “Figure 3-1 … to typical components.”
2014-03-21	4A.4	CCB-47, row 15: Inserted “ordering” into first sentence of the third paragraph.
2014-03-21	6A.1	CCB-47, row 16: Changed heading from “7-bit ASCII …” to “ASCII …”. Changed first sentence to “The ASCII character set is the encoding of characters into 7-bit integer numbers according to the ANSI standard (see Section 1.5):”
2014-03-21	4C.1	CCB-47, row 17: From sub-paragraph 2, removed “,as required for the multipurpose transport of files (RFC 2046; http://www.rfc-editor.org/info/rfc2046)”
2014-03-21	5A	CCB-47, row 18: Removed the first sentence in the second paragraph; appended the remainder of paragraph 2 to paragraph 1.
2014-03-21	5A.3	CCB-47, row 19: Removed “and PDS types based on these” from the first sentence.
2014-03-21	Table 5C-3 (now Table 5A-3)	CCB-47, row 20: In second column of row for ASCII_Integer changed “decimal integer” by “signed integer in base 10”.
2014-03-21	Table 5C-3 (now Table 5A-3)	CCB-47, row 21: Changed second column of row for ASCII_NonNegative_Integer from “a decimal integer” to “an unsigned integer in base 10”
2014-03-14	5B	CCB-47, Row 22: Swapped the order of the two sentences in paragraph 2 for improved clarity.
2014-03-21	5C.2	CCB-47, row 23: Changed “the internal IEEE floating-point representation of real numbers” to “internal IEEE floating-point representation of real numbers”

Date	Section(s)	Change(s)
2014-03-21	5C.3	CCB-47, row 24: Add a new first sentence “This section describes the internal IEEE floating-point representation of complex numbers.”
2014-03-21	5C.3	CCB-47, row 25: Changed “two IEEE Real format numbers of the same precision, contiguous” to “two IEEE-format real numbers (Section 5C.2) of the same precision, which are contiguous”
2014-03-21	5C.4	CCB-47, row 26: Changed “do not align” to “may not align” in the first sentence.
2014-03-21	6B	CCB-47, row 27: Changed first sentence in paragraph 2 from “There is a ‘common’ namespace managed by EN (under the steward name 'pds'), and each PDS discipline node manages its own namespace.” to “There is a common namespace managed by EN (under the steward name 'pds'); each PDS discipline node also manages a namespace.”
2014-03-21	6B	CCB-47, row 28: Added period at end of third paragraph.
2014-03-21	6B.3	CCB-47, row 29: Appended “(see (W3C, 2012b))” to first bullet
2014-03-21	6C	CCB-47, row 30: Added reference to Section 6C.1.1 after “base name” in second paragraph.
2014-03-21	6C	CCB-47, row 31: Reworded final paragraph.
2014-03-21	6C.1	CCB-47, row 32: Removed heading for Section 6C.1.2, decremented numbers of all following subsections of the same order, and corrected references to these subsections elsewhere in the document.
2014-03-21	6C.2.3	CCB-47, row 33: Copy the contents of 6C.1.4 to a new Section 6C.2.3 “Prohibited Directory Names”; add “core” to the list.
2014-03-21	6E.1, 6E.2	CCB-47, rows 34 and 35: Made bullets have consistent format.
2014-03-21	8A.2.2	CCB-47, row 36: Changed “TeX/LaTeX” to “LaTeX” in next to last paragraph.
2014-03-21	8B.2	CCB-47, row 37: Changed “logical identifier (LID)” to “versioned identifier (LIDVID)” in paragraph 3.
2014-03-21	8C	CCB-47, row 38: Dropped sentence “PDS requires that raw data be included in archives and that the calibration steps subsequently applied be documented.” There is no PDS requirement for raw data.

Date	Section(s)	Change(s)
2014-03-21	Tables 5C-1 through 5C-4	CCB-47, row 39: Renumbered tables to be 5A-1 through 5A-4; updated references to tables throughout the rest of the document.
2014-03-21	Table 6B-1	CCB-47, row 43: Corrected URI for Mars Pathfinder and Namespace ID and URI for LADEE.
2014-03-21	Table 6B-1	CCB-47, row 44: Added example of namespaces within mission namespaces.
2014-03-21	1.5	CCB-47, row 47: Added RFC 3629 to the list under Requests for Comment.
2014-03-21	7E	CCB-47, row 54: Replaced second paragraph with more explicit instructions.
2014-05-21	6A.4	CCB-47, row 55: Changed “7-bit ASCII” to ASCII in newly added 6A.4.
2014-03-26	2A.4	CCB-54: Changed “a different collection. A product’s member status (primary or secondary) is based on its *first* association with the collection” to “an additional collection. A product’s member status (primary or secondary) is based on its *first* association with a collection”
2014-03-26	2A.5	CCB-54: Paragraph about *document* collections: Changed “closely related documents” to “related documents”.
2014-03-26	2A.5	CCB-54: Paragraph about *miscellaneous* collections: Removed “a table of updates to product label values made after the bundle has been ingested by PDS may be included in a miscellaneous collection. Other examples include” and added “could be included after “… modification history”.
2014-03-26	4	CCB-54: Second sentence: Changed “All data products” to “All products”.
2014-03-26	4D	CCB-54: End of first paragraph: Change “data” to “digital objects”
2014-03-26	4D	CCB-54: Second paragraph: Generalize the first sentence from “There are three subclasses …” to “The subclasses of … include …” Append sentence: “Encoded_Byte_Stream is not an option for Product_Observational.”
2014-03-26	2A.1	CCB-54: Replaced “it” by the actual antecedent in the third and fourth paragraphs.
2014-03-26	3	CCB-54: Replaced “an XML schema document (XSD)” by “XML schema document(s) (XSD) and XML Schematron document(s) (SCH)”

Date	Section(s)	Change(s)
2014-03-26	6B.2	CCB-54: Removed a stray comma at the end of the section.
2014-03-26	6D.3	CCB-54: Replaced “are” with “must be” in three places.
2014-03-26	6D.3.2	CCB-54: Replaced “is” by “must be” in second paragraph.
2014-03-26	9C.2	CCB-54: Updated example Inventory definition to show title case enumerated values, and bolded offset statement to show that it is required verbatim.
2014-03-27	1.4	CCB-54: Add sentences mentioning Sections 9 and 10.
2014-03-26	Tables 2B.1, 2B.2, 6B.1	CCB-54: Change to Tables 2B-1, 2B-2, and 6B-1
2014-03-26	Figures 2B.1, 2B.2, and 2B.3	CCB-54: Change to Figures 2B-1, 2B-2, and 2B-3.
2014-03-27	10	CCB-55: Added section containing PDS policies which pertain to archives and the archiving process.

Version 1.3.0

Date	Section(s)	Change(s)
2014-09-10	10	CCB-79: Removed text, substituted pointer to EN web site with PDS policies.
2014-09-16	2B.1.1	CCB-57: Added third paragraph requiring co-location of labels and associated digital objects.
2014-09-17	8A.2.2	CCB-80: Removed text about formats of document files (most of the last four paragraphs).
2014-09-17	8A.3	Typo: at end of second paragraph Product_Documentation corrected to Product_Document.
2014-09-18	9E	CCB-46: added section on Product_Native constraints

Version 1.4.0

Date	Section(s)	Change(s)
2015-06-02	4A.4	CCB-102: Changed requirement to one (and only one) Axis_Array per dimension for the Array class and subclasses in the final paragraph.

Date	Section(s)	Change(s)
2015-06-02	9F	CCB-98: Added new section on class selection.
2015-06-28	2B.2.2.2 Table 2B-2 Figure 2B-1 Figure 2B-2 Figure 2B-3 6C.1.3	CCB-117: Changed collection[_*].tab to collection[_*].csv and collection.tab to collection.csv
2015-06-02	1.4	Added final sentence about Section 10, which provides links to PDS policy statements.
2015-07-27	2B.1.1 6C.2.4	Revised third paragraph in 2B.1.1 to allow directory_path_name to be used only for a subdirectory relative to the label. Added 6C.2.4.
2015-07-27	Figure 2B-2	Removed duplicate PDS4_PDS_0300b.xml; updated examples from 0300 to 1400
2015-06-02	2B.2.2.2	Corrected errors in text references to figures and tables. P. 20: “Figure 2B.3” should be “Figure 2B-3” P. 55: “Table 6B.1” should be “Table 6B-1”
2015-06-02	4C	Dropped “or a line-feed by itself” from the first paragraph.
2015-06-02	5A.2	Corrected DOY date range from “01-365” to “001-365”
2015-06-02	Table 5A-2	Corrected footnote “6” to “7”
2015-06-02	5C.3	Removed duplicate “in memory” from third line.
2015-06-23	Table 6B-1	Deleted lde; LADEE never created a mission namespace.
2015-06-02	9B	Added angle brackets around “fields” and “groups”
2015-06-02	9E	Added angle brackets around “archive_status” in next to last paragraph.
2015-07-09	8A.2.2	Added text about specific document formats to final paragraph (wording suggested by Joyner based on PDS policy on acceptable formats, modified slightly by Gordon). Then split off final sentence into a paragraph of its own.
2015-07-09	9E	Revised the wording of paragraph 3 for improved clarity.
2015-07-28	1.5	Corrected URL to http://tools.ietf.org/html/rfc3629
2015-07-28	2A.1	Corrected “LIDs/LIDVIDS” to “LIDs/LIDVIDs”

Date	Section(s)	Change(s)
2015-07-28	4A.1	Changed “and the second index varies next to slowest.” to “and the first index varies slowest.”
2015-07-28	4A.5	Changed “screen” to “display device” (twice).
2015-07-28	4C.2	Change “Field delimiters (if any)” to “Field delimiters”
2015-07-28	6C	Changed “255 character names” to “names with 255 characters”
2015-07-28	6D.3.1 6D.3.2 6D.3.3	Corrected numbering; formatted as Header3
2015-07-28	6D.3.2 6E.1 6E.2	Added missing periods at end of sentences
2015-07-28	6E.3.2	Changed “Units_of_Measure_Length” to “Units_of_Length”
2015-07-28	6E.3.2	Added “km” to the list of possible choices
2015-07-28	7A	Changed “Unit_of_Measure_” to “Units_of_” in the three classes at the end of paragraph 1.
2015-07-28	9C	At the end of the first line changed “delimited records” to “delimited records, called an Inventory table,”
2015-07-28	9C.2	Corrected indent for second </Field_Delimited>
2015-08-10	2B.2.2.2	Third paragraph: Removed “of the data providr’s choosing” and to substituted “name” for “prefix”.
2015-08-10	3	Appended new sentence describing File_Area_Definition to the paragraph before Figure 3-1.
2015-08-10	4A.4	Deleted last sentence in paragraph 1: “The number of axes in a scientific data file can exceed three, but it rarely does.”
2015-08-10	4A.4	Added new next-to-last paragraph: “Particle data often have one or more look directions (for example, polar and azimuthal angles) and an energy dimension. While these data have similarities to ‘band sequential’ formats, they are not image data in the traditional sense.”
2015-08-10	4B.1.2	First paragraph: changed “can be used” to “is to be used”.

Date	Section(s)	Change(s)
2015-08-10	4B.1.2	In the paragraph about "[+|-]": changed "In PDS4 tables, the" to "The".
2015-08-10	4C	Next to last paragraph: Added "CDF" to the list of header formats recognized as parsable bytes streams.
2015-08-10	4C.2	Changed "several attributes" to "attributes".
2015-08-10	5A.2	Reorganized bulleted list so that the colon, hyphen, and upper case "T" constraints are separate items.
2015-08-10	8	Changed "data mean" to "data represent".
2015-08-10	8A.2	Combined last two sentences.
2015-08-10	8A.2.1	Deleted first paragraph and first sentence of second paragraph. Added new lead sentence and edited the next.
2015-08-10	8A.2.2	Removed "PDS seeks to err on the side of completeness" and substituted "completeness of information is the goal".
2015-08-10	8A.2.2	Removed final sentence from second paragraph: "Documents which contain spelling errors, poor grammar, or illogical organization will be rejected and may ultimately lead to rejection of the submitted data."
2015-08-10	9C	Added "(primary or secondary)" after "member status".
2015-08-10	9C.1	Second paragraph: reworded the first sentence; added a new last sentence.
2015-08-10	9F	Corrected wording to include Array_2D as one of the possible choices.
2015-08-19	Various	Changed "section" to upper case where it referred to a numbered section elsewhere in the document.
2015-08-19	Table of Contents	Manually corrected error in page numbers 13-80 caused by unknown problem.
2015-09-22	Change Log	Corrected typo in Change Log entry 2015-08-10 8A.2.2.

Version 1.7.0

Version 1.7.0 of the Standards Reference incorporates changes made in the Information Model versions 1.5.0, 1.6.0, and 1.7.0. There are no corresponding versions 1.5.0 and 1.6.0 of the Standards Reference because of a delay in updating the document. To keep the Standards

Reference version synchronized with the Information Model version, the Standards Reference version was incremented from 1.4.0 to 1.7.0.

Date	Section(s)	Change(s)
2016-07-15	Table 5A-2	CCB-101. Removed ASCII_Date, ASCII_Date_Time, and ASCII_Date_Time_UTC. Added ASCII_Date_Time_YMD_UTC and ASCII_Date_Time_DOY_UTC.
2016-07-15	6D.2	CCB-120. Added example of non-PDS agency LID (urn:esa:psa)
2016-07-15	4B	CCB-114. Revised description of table data to remove implication that binary records may have record delimiters.
2016-07-15	9C.3	CCB-129. Added section to list constraints on mission science data collections.
2016-07-15	2A.1, 6C1.3	CCB-139. Changed description of readme file from text or HTML to 7-bit ASCII or UTF-8 text (no HTML).
2016-07-15	6D.3.1	CCB-145. Added explanation of when versions must be incremented.
2016-07-15	4B.1.2	CCB-152. Revised definition of field_format.
2016-07-26	5A.2	CCB-161 Revised text to state that date and time fields and attributes must be defined to use only one of the two available formats, calendar (YMD) and ordinal (DOY).
2016-07-26	Change log	Moved change log from beginning to end of document.
2016-07-26	All	Corrected formatting and style inconsistencies (not change-tracked).
2016-07-26	6	Added missing section 6 heading. Could find no previous version in which the heading was not missing, so gave it the title "Naming".
2016-08-02	6C.1.1, 6C.2.1	Reworded text to clarify uniqueness requirements for file and directory names, in response to request from NSSDCA.
2016-08-15	2A.5	Mentioned "other responsible agency" in addition to PDS EN as curator of context products, in response to IPDA comment. Removed reference to XML Catalog in archive bundle.
2016-08-15	2A.5	Added Product_Ancillary to list of examples of products in a geometry collection, in response to IPDA comment.
2016-08-15	2B.2.2.2	Referred to example of directory structure on PDS4 Examples web page, in response to IPDA comment.

Date	Section(s)	Change(s)
2016-08-15	3	Added statement that international agencies control changes to their own schemas without the need for PDS approval, in response to IPDA comment. Added fully resolvable namespace and schema location to PDS4 repository.
2016-08-15	6B.1	Added statement that EN creates namespaces in consultation with the discipline nodes or with the corresponding international authority, in response to IPDA comment.
2016-08-16	4A.1, 4A.2	Clarified labeling of arrays stored in Last_Index_Fastest and First_Index_Fastest order, and corrected statement of ISIS use of First_Index_Fastest storage order, in response to IPDA comment.
2016-08-16	Table 6B-1	Added "/v1" to each URI in the table; added a bullet to the formation rules "end with 'vN' where N is a version number. In the examples in Table 6B-1 N is 1". Done in response to IPDA comment.
2016-08-16	6D.3.1	Reworded last paragraph to clarify when collection and bundle VIDs must be incremented, in response to IPDA comment.
2016-08-16	8A.2.2	Changed "Software Interface Specifications (SISs)" in list of example document products to "Archive content, format, and organization, e.g. Software Interface Specifications (SISs)" in response to IPDA comment.
2016-08-16	8B	Added statement "Context products in a namespace belonging to an international agency are controlled by the steward of that agency" in response to IPDA comment.
2016-08-16	All	URLs throughout the document were inconsistent; some were live links, some were not. For consistency all URLs have been unlinked. For example see Table 6B-1.
2016-09-29	6B	Added footnote to first sentence in second paragraph to mention an exception to the rule that everything in the common namespace is managed by EN; the exception is Instrument.Subtype. Suggested by Steve Joy as a result of resolution of CCB-163.
2016-09-29	1.3	Added paragraph stating that while the document is written in the context of NASA PDS archives, the PDS4 standards may be used by other agencies. To explain use of "other responsible agency" in section 2A.5 para. 4 and elsewhere, in response to Document Reviewer comment.

Date	Section(s)	Change(s)
2016-09-29	6E.3	Simplified awkward sentence at end of first paragraph, in response to Document Reviewer comment.
2016-09-29	8A	Clarified statement about additional processing, in response to Document Reviewer comment.
2016-09-29	9D	Re-worded first sentence to explain 'read me' file, in response to Document Reviewer comment.
2016-09-29	4B	Changed "associations are so similar" to "associations are very similar" in paragraph 4, sentence 4, in response to Document Reviewer comment.
2016-09-29	5A.2	Changed "decimal seconds" to "decimal fractional seconds" in definition of "fff", in response to Document Reviewer comment.
2016-09-29	1	Made the 1 at the end of the first paragraph a superscript.
2016-09-29	3	Changed "XML Schematron" to "Schematron" in response to Document Reviewer comment.
2016-09-29	2A	Referred the reader to the Concepts document for definitions of terms such as bundle and collection, in response to Document Reviewer comment.
2016-09-29	2A.2	Changed "Data are organized into collections" to "Basic products are organized into collections" in response to Document Reviewer comment.
2016-09-30	2A.3	Italicized "aggregate products" for consistency, in response to Document Reviewer comment.
2016-09-30	2A.5	Removed stray yellow space, in response to Document Reviewer comment.
2016-09-30	Table 2B-1	Removed HTML as an option for the readme file, in response to Document Reviewer comment, and updated text accordingly.
2016-09-30	Fig. 3-1	Inserted missing <CR> between Primary_Result_Summary and File_Area_Definition, in response to Document Reviewer comment.
2016-09-30	5C.1.3	Corrected text to say that the 1-byte signed integer is shown in figure 5C-4, not 5C-12, in response to Document Reviewer comment, and corrected superscripts
2016-09-30	6D.3.1	Added missing period to last sentence on page 64, in response to Document Reviewer comment.

Date	Section(s)	Change(s)
2016-09-30	5C.1.4	Corrected figure citations, in response to Document Reviewer comment.
2016-09-30	9C.2	Restored missing bold formatting to example label, in response to Document Reviewer comment.
2016-09-30	9C3.1	Removed errant quotation mark.
2016-09-30	Change Log	Corrected entry for 2016-07-15, CCB-129, which had the wrong CCB number, in response to Document Reviewer's request.

Version 1.8.0

Date	Section(s)	Change(s)
2017-03-21	Table 5A-3	CCB-171. Revised definitions of ASCII_Integer, ASCII_NonNegative_Integer, and ASCII_Real.

Version 1.9.0

Date	Section(s)	Change(s)
2017-08-09	Table 5A-3	CCB-165. Revised definitions of ASCII_Numeric_Base2, ASCII_Numeric_Base8, and ASCII_Numeric_Base16.
2017-08-09	8A.2.2	CCB-172. Added paragraph about video format files.
2017-08-09	Table 6B-1	CCB-176. Updated names of Cartography and Imaging Sciences Node and Ring-Moon Systems Node.
2017-08-09	6D.1	CCB-178. Added sentence "The local identifier value must start with either a letter or an underscore."
2017-08-09	6B	CCB-180. Added "Only namespaces that are registered in the Namespace Registry (https://pds.nasa.gov/pds4/schema/pds-namespace-registry.pdf) are allowed in PDS labels." Did not add the requested sentence "To add namespaces to the Registry consult the Namespace Registry Document." because this document does not tell how to add a namespace.
2017-08-09	4C.1	CCB-181. Added "<CR> and <LF> may be used only in combination as a record delimiter. They may not be used, either separately or in combination, as part of a field value."

Date	Section(s)	Change(s)
2017-08-22	9G (new)	CCB-192. Added new section Product_Metadata_Supplemental.

Version 1.10.0

Date	Section(s)	Change(s)
2018-03-02	5C	CCB-153. Updated descriptions and illustrations of binary data types and expanded descriptions of bit strings. Added 5C.4.1 and 5C.4.2. Included reference to PDS policy on use of packed data at beginning of 5C.4.
2018-03-05	9H (new)	CCB-174. Added description of Composite_Structure class.
2018-03-05	9C.3.1, 9D.3.1 (new)	CCB-189. Revised constraints on Mission Science Data Collection. Added constraints on Mission Science Archive Bundle.

Version 1.10.1

Date	Section(s)	Change(s)
2018-05-10	Fig. 3-1, 8A.3	CCB-177. Added example and description of Source_Product_Internal and Source_Product_External classes.

Version 1.11.0

Date	Section(s)	Change(s)
2018-09-14	9C.2, 9D.2	CCB-206. Added requirement that author_list or editor_list must be present in Citation_Information if the doi attribute is present.
2018-09-14	Table 5A-3	CCB-213. Changed permitted values for ASCII_Real.
2018-09-18	5A.2	CCB-226. Changed definition of fractional seconds to say “one or more digits commensurate with precision”. Changed examples to show six digits of precision instead of three.

Version 1.12.0

Date	Section(s)	Change(s)
2019-04-01	1.5, 5A (Table 5A-4), 5B (Table 5B-1)	CCB-235. Added data type "ASCII_BibCode" to table of String Types and Character Data Types. Added reference to BibCode documentation at the Astrophysics Data System.
2019-04-01	4B.1.2	CCB-214. Added <validation_format> to discussion and formation rules for field formats in a character table.

Version 1.13.0

Date	Section(s)	Change(s)
2020-01-08	1.6, 2B.2.2.2, 5C.4, 6B, 10	Updated links to documents on reorganized PDS web site.
2020-01-08	1.4	Removed "They are usually implemented through Schematron files" because the concepts of schema and Schematron have not yet been introduced (reviewer comment).
2020-01-08	2A.5	Added footnote about schema and Schematron files.
2020-01-08	2A.1	Replaced reference to LIDs/LIDVIDs with "unique identifier" because LID[VID] is not defined yet (reviewer comment).
2020-01-08	2A, 3, 4B.1.2, 5, 8, 9	Minor edits from reviewer comments.
2020-01-08	4A.4	Added text to clarify meaning of 3D array storage terms, in response to reviewer comment.
2020-01-08	3, Fig. 3.1	Put label components in correct order in Figure 3.1 and in text above it (reviewer comment).
2020-01-08	2A.3	Changed "RGB" to "red, green, and blue" (reviewer comment).
2020-01-08	6D.3	Changed "subsequent fields" to "fields n, o, and p" (reviewer comment).

Version 1.14.0

Date	Section(s)	Change(s)
2020-04-14	8A.3	CCB-220. Revised to explain use of a table of source products, using text provided on CCB-220 Jira page.
2020-04-14	9F	CCB-280. Revised text regarding Most-Specific-Class requirement, using text provided on CCB-280 Jira page.
2020-04-28	6B.3	Added note in response to reviewer's misunderstanding of namespace URIs: "It is important to note that while these namespace URIs look very much like URLs (Universal Resource Locators), as they begin with 'http:', they are not actual addresses of Internet web pages and are not meant to be used in web browsers."
2020-05-13	2B	Added text to make clear that the directory structure described in Section 2B is given as an example only, not as a requirement. Reviewed by Steve Hughes (EN), Jordan Padams (EN), Joe Mafi (PPI), Mitch Gordon (RMS), Boris Semenov (NAIF), Ed Guinness (GEO), Trent Hare (CIS), and R. Chen (EN). This is a temporary fix, as the DDWG intends to rewrite this section.
2020-05-22	2A.5, 2B.1.1, 2B.2.2, 4A, 5A.2, 5C.4	Minor edits for clarity by T. McClanahan

Version 1.15.0

Date	Section(s)	Change(s)
2020-10-02	4D	CCB-289. Added Encoded_Audio to list of subclasses of Encoded Byte Stream.
2020-12-04	5A.1, 5A.2, 5C.3	Added references to tables and figures in text, in response to a reviewer's comment

Version 1.16.0

Date	Section(s)	Change(s)
2021-04-21	4C.1, 8A.3, 9C.1	CCB-264. Allow line feed alone as a record delimiter.

Version 1.17.0

No changes were made to the Standards Reference for Information Model version 1.17.0.

Version 1.18.0

Date	Section(s)	Change(s)
2022-03-30	9G, paragraph 2	CCB-343. Revise Product_Metadata_Supplemental.
2022-03-30	2A.3, 2B.2.2.1, 2B.2.2.2, 3, 6C.1.3, 6C.1.6	CCB-260. PDS4 Documentation Changes for Label Extenstion (enclosure)