The Planetary Science Journal, 2:204 (12pp), 2021 October https://doi.org/10.3847/PSJ/ac1f22

The Road to an Archival Data Format—Data Structures

Anne Raugh[1] and J. Steven Hughes[2]
[1] Department of Astronomy, University of Maryland, College Park, MD 20742-2421, USA; araugh@umd.edu
[2] Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA
Received 2020 September 21; revised 2021 August 8; accepted 2021 August 17; published 2021 September 28

Abstract

The current data formatting and labeling standards for the Planetary Data System (PDS) are known as the PDS4 Standards. They supersede the PDS3 Standards, but they represent a complete redesign of the requirements and implementation, rather than even a major incremental revision, from the previous standard. At the heart of the PDS4 Standards lies a fundamental, philosophical change from the PDS3 paradigm: the PDS4 Standards clearly and specifically constrain the way that the bytes comprising observational data may be stored in their data files—that is, the data structures—to a much greater degree than the PDS3 Standards ever did, even in their most mature realization. In PDS4, the PDS has defined data structures optimized for the long-term preservation of observational data. We explore the history of the PDS and its standards through the examination of a single, simple data structure (the 2D image), to understand the evolutionary pressures on the data and on the PDS that led to the development of the archival data structure requirements for observational data at the core of the PDS4 standards.

Unified Astronomy Thesaurus concepts: Astroinformatics (78)

1. Introduction

File formats are essentially functional. The typical contemporary file format combines a data structure with metadata providing structural details (data types and byte order, for example), processing history, geolocation data, and so on. The same logical content can be formatted in myriad ways, and which way is "best" is a primarily subjective assessment. The "best" format might be the one that a preferred software tool reads and writes, or the format that is small enough to fit 100,000 files on one thumb drive, or the format that preserves all the depth and precision of the original source data. Each environment, each application has its own criteria for "best," or even just "usable."

The Planetary Data System (PDS) has been operating since 1990, charged with preserving planetary mission data in a usable form for future generations. The question of the "best" format for PDS data has been and still is a source of constant debate. The original PDS design took the view that the "best" format was the one the mission scientists used, and that PDS should bridge any gap for nonmission users via software, documentation, and support. That view, however, evolved—along with every other aspect of information technology—over the first two decades of PDS operations.

In this discussion, we focus on the lowest-level starting point for "PDS format"—the storage structures; we examine the effects of changing technology, budgets, and user expectations on the data structures described in the original standards; and we define the characteristics of the data structures laid out in the new PDS4 standard[3] that make them specifically suited for archiving observational data.

Note, however, that the data products composing the whole of the PDS archive are many and widely varied: images, maps, spectra in every wavelength regime, particle counts, EM field measurements, radar shape models, laser ranging measurements, Deep Space Network tracking files, gravity models, atmospheric experiments, and so on. Not surprisingly, each discipline specialty has its favorite tools and preferred file formats for analysis and archiving. In order to focus on the issues at the core of the PDS data structure evolution rather than the details of specific file formats from various disciplines, we will make a case study of the simplest data structure in the PDS archives: 2D raster images. The complications arising even with this generally well-understood structure illustrate the problems found in more complex structures and their related file formats.

We start by placing the development of the PDS and its standards in historical context.

2. A Brief History of the PDS

The PDS was established to ensure that data collected by and essential to NASA's planetary missions became part of a national resource—a high-quality research archive that would serve the needs of specialist and interdisciplinary researchers now and for the indefinite future. The PDS was created in response to what was widely perceived as a data crisis:

> "It is noted with increasing alarm by many in the science community that valuable data sets are disappearing. Some become lost because of deterioration of the media upon which they are stored. Some sets are effectively lost because the documentation was not retained or the software required to read and interpret the data no longer compiles on current computer systems. In a few cases, the knowledgeable individuals (of the data set) have left the field through career changes, retirement, or death. Loss of data knowledge can be expected to

[3] Planetary Data System (2021). Note that the PDS4 Standards Reference is kept synchronized with the Information Model and reissued in a parallel version with each biannual release. The current version is posted at https://pds.nasa.gov/datastandards/documents/sr/current.

accelerate in the next few years unless appropriate action is taken."[4]

2.1. CODMAC1 and the Planetary Data Workshop

In 1978, The Space Science Board formed the Committee on Data Management and Computation (CODMAC) and charged it to investigate the state of data resulting from spacecraft observations, to identify problems, and to make recommendations. The first of several resulting reports (National Research Council 1982) did precisely that, laying out seven "Principles for Successful Scientific Data Management."[5] This, in turn, seeded the creation of a number of data system projects, among them the Pilot PDS (sponsored by NASA Code EC) to investigate the technological aspects of data curation and distribution, and the PDS project (sponsored by NASA Code EL) to work out the user requirements. The Planetary Data Workshop was held at Goddard Space Flight Center in late 1983 to assess the current state of planetary data and make recommendations for the nascent PDS (Kieffer et al. 1984). In 1985 the two NASA projects were merged under the PDS rubric, and development proceeded in earnest.

2.2. The PDS—Original Concept

The original concept and design for the PDS was for a central archiving node and distributed discipline nodes.[6] The central node would be the permanent repository for the archival data, the location of the high-level catalog used for finding data within the system, and the primary physical distribution center for whole data sets. Users would log into the catalog system, select data sets based on attributes stored in the high-level catalog, and then request that copies of the selected data either be provided or be directed to a discipline node for further assistance. Note that this was the mid-1980s. "Distribution" was expected to involve making copies of reels of magnetic tape and shipping them over land to the recipient for large volumes of data, although the version 1.0 specification did include tasks to develop electronic data transfer capabilities, where that was feasible, and lists "CDROM" as a desired output format for PDS data.

The central node would also develop software to support activities at the discipline nodes, including the software for the detailed catalog databases and interfaces, data format transformation software, and analytical software. The software would be developed centrally and then deployed to each discipline node to provide a uniform user experience.

The discipline nodes were conceived as centers of research, managed by scientists with a research interest in the data they were supporting. Discipline nodes would maintain detailed catalogs for the data of interest to their discipline clients and would have local copies of the data—possibly in a more computationally convenient format than the archive format—that could be used directly with locally hosted software to perform some level of analysis. The discipline nodes would be able to distribute small quantities of data, would advise users who needed assistance selecting or understanding data holdings, and would also be involved in creating archive submissions.

2.3. Requirements Reviews

A review of the PDS system requirements was held in 1986 July. The proposed time line for development contained a three-stage rollout over the course of 5 yr, with the stages identified as PDS V1.0, PDS V2.0, and PDS V3.0, with PDS V3.0 being the final, fully operational system of Central Node (at JPL) and discipline nodes selected by competitive proposal.

Significantly, the final presentation of that 1986 review was by Project Manager J. T. Renfrow, who noted in his "Issues and Concerns" that "the current scope of the complete statement of the functional capability for Version 1.0 does not match the resources (schedule and dollars) available."[7] He then lists nine major issues related to this topic alone, ranging from scope of various software development efforts to division of responsibilities between Central and discipline nodes, and a recognition that "node personnel are already overextended and are not contractually committed to PDS system software development."[8]

Requirements development continued, and a Functional Requirements Review was held, most likely in 1987–88. Surviving documents from that review could not be found, but the community feedback from that review had an influence on changes to the original concept. The community recommended a change from the centralized archive/distribution hub with discipline research centers to a distributed archive model where the discipline nodes were the primary data curators for their discipline data, in addition to supporting end users and the creation of new archive data sets. In this model Central Node would be the first-line archive backup location for all nodes[9] and would develop system software for use by all discipline nodes. Note that this shifted some of the burden for software development, in particular for discovery and distribution, to the nodes.

2.4. PDS1, PDS2, PDS3

The system delivery review for the PDS V1.0 operational release was held in 1990 February, the discipline nodes having been selected about 5 months previously. The Version 2.0 release followed shortly after, in 1992. The Version 3.0 release came in 1994 and included some modifications to cataloging and labeling requirements, resulting from the early node experiences with archiving legacy data and working with new data providers for missions in development. This last release came to be known as "PDS3" for the acronym used as a version identifier in the corresponding labels. The last minor revision of the PDS3 standards was version 3.8, released in 2009 February (although the data dictionary database is even now updated regularly for the PDS3 data sets still in grandfathered production).

Despite the resource shortfalls noted by Renfrow in 1986 and the redesign work needed to respond to the community feedback, the PDS3 released in 1994 retained the same basic outline as the original concept. There was a Central Node supporting the work done at the discipline nodes, for example, and there was a basic documentation set and supporting software. What was lost were the additional tiers of software

[4] Kieffer et al. (1984, p. 1).
[5] National Research Council (1982, pp. 6-7).
[6] Planetary Data System Design Team (PDS-DT) (1986, p. 10).
[7] Renfrow et al. (1986, p. 194).
[8] Renfrow et al. (1986, p. 195).
[9] In 1986 the PDS and the then NSSDC signed a Memorandum of Understanding establishing the NSSDC as the deep archive for all PDS data.

support originally envisioned—the format transformations and analytical tools.

2.5. PDS3 Format

It is important to note that "PDS3 Format" is a bit of a misnomer. The designers of the original standards were developing a labeling language that could describe existing file structures and augment those with additional metadata, rather than creating a new format "from scratch." The labeling language, created at JPL, was called Object Description Language (ODL), where "object" was in reference to the data objects of object-oriented programming, a rising technology at the time. ODL had applications beyond PDS, because it was essentially a parsing standard (like XML), defining the keyword=value syntax, the OBJECT statement for grouping, the END statement to signal the end of a label, and so on. It was up to the application environment to develop the keywords comprising the project-dependent metadata. The PDS designers developed the Planetary Science Data Dictionary (PSDD) to define the keywords that would be used in PDS labels. In PDS labels the keywords were constrained—all keywords appearing in PDS labels had to be formally defined in the PSDD.

The PSDD also contained OBJECT definitions—sets of keywords that were used to provide input/output parameters and other metadata related to the structure and content of the data objects. The PSDD listed required and optional keywords for each OBJECT. Objects could be nested inside other objects, and so it was possible to define hierarchical metadata to describe the storage format of any file a planetary mission was likely to produce. And if the PSDD could describe the data object structure, then software should be able to parse out the necessary information from the label to read the data. (The canonical validator software, lvtool, referenced a flat-file version of the PSDD to validate label content).

This descriptive approach to labeling was in keeping with the comments regarding data format in the documents that led to the creation of the PDS. The National Research Council (1982), for example, mentions data format in its "Principles for Successful Data Management":

> "The data formats should strike a balance between flexibility and the economies of non-changing record structure. They should be designed for ease of use by the scientist."[10]

The second CODMAC report (National Research Council 1986) talks about archives containing "basic science data in various forms" accompanied by "the basic software tools needed to access the data."[11] Kieffer et al. (1984) include a section on "Standardization Recommendations for Digital Data" that suggests an approach similar to the Flexible Image Transport System (FITS) and Video Image Communication and Retrieval (VICAR) System formats for archive data—that is, a format consisting of a standardized header describing the data structure that follows, recognizable to processing software.

These existing file formats, which themselves combined metadata with data structures, served as the models for the PDS3 label development. The initial collection of defined data object structures was based on archetypes from the first data sets added to the archive, and they were defined with the intention of being sufficiently flexible to cover the data structures found in any file format likely to be created by a mission science team.

Although we will focus on images, these first data structures also included ASCII and binary tables, text files for documentation, and a few other formats to support the more complex data types returned by instruments other than simple framing cameras.

2.6. The Object Access Library (OAL)

A software library was produced internally for use in standardizing PDS label and file operations as part of the design and development phases of PDS V1.0. During the early operational years of PDS, this code was adapted to V2.0 and V3.0 and was ultimately released for public use in 1995 as the Object Access Library (OAL), providing read and write access to label keywords and data objects.

The OAL was the basis for the PDS data display tool NASAView and the canonical validator for PDS labels, lvtool. As the core library of the PDS services, the OAL was essential to archive verification and validation. The ability of the PDS to maintain usability of the data depended critically on the OAL and its successors being able to read every label in the PDS archive and, among other things, use the data structure description therein to access the data.

The OAL was released as a precompiled, shareable object library; users needed to link it into their own compiled source in order to use its subroutines. The OAL, in turn, was based on an ODL parsing library known as the Label Library Lite (L3), which was also released as a shareable object library.

3. The Early PDS3 ERA, 1990–2000

By 1995, then, the PDS3 Standards Reference (SR) defining the PDS archive content requirements and labeling rules was fully fleshed out, the PSDD provided keyword content, NASAView provided visualization, lvtool provided label validation, and the OAL and L3 libraries were available to support users and developers who wanted to code support for PDS3 archive files. But even from these early days, PDS was hearing complaints from users about the proliferation of "data formats" within the archive, and developers showed little interest in providing PDS3 support the way they had a decade earlier for the FITS format. As years passed, the format complaints became louder, few if any end users made use of the OAL or L3 libraries, and third-party support for PDS3 labels was conspicuous by its absence.

The FITS and VICAR formats mentioned as models by Kieffer et al. (1984) had already enjoyed over two decades of success in their respective user communities when they were cited as models for PDS labeling standards. They were both essentially image formats (FITS had not yet added table structures to its standard extensions list). VICAR was a format named after the software processing environment that produced it, while FITS was a transport format that required read/write support be written for each processing environment into which it was incorporated.

As of this writing, 35 yr after Kieffer et al. (1984) held them up as role models for the PDS standard, the FITS and VICAR formats are still in use—but the PDS3 format was found to be untenable and retired after barely 20 yr. To understand why, we

[10] National Research Council (1982, p. 6).
[11] National Research Council (1986, p. 36).

need to understand how and why the PDS3 standards were regularly modified and consider the sea changes in attitudes and perceptions of users to the nebulous concept of data format. Given the simplicity of the basic image data structure and the fact that the model formats for PDS were both image formats, we will focus on the issues arising with the IMAGE data structure to characterize the problems encountered with the wider range of PDS data structures.

3.1. A Brief Aside—Displaying an Image

Before digging into the details of image formats, it is worth remembering the programmatic actions necessary to read an image from a file and display it on a device. There are two distinct processes that happen in succession: first, the data are read from the file into program memory; second, the elements of program memory are mapped to pixel locations on the display device. File format definitions deal only with the first process—that is, how a sequence of bytes is read from the file and interpreted into program memory. This results in a section of memory that is then treated as a two-dimensional structure following the conventions of the software environment. It is the software environment definition, often adjustable through user settings, that defines how that program memory is mapped onto a display device. In particular, it is the software environment that determines whether the (1,1) pixel of an image is located in the upper left corner or the lower left corner on the display.

3.2. The First Signs of Trouble

The first data sets archived in the new PDS were legacy data sets from recent planetary missions: Viking (the very first), Voyager 1 and 2 (which are still returning data), Magellan, and Galileo. Each of these spacecraft hosted a variety of instruments, including imagers. The image data sets were all produced by the Multimission Image Processing Laboratory (MIPL), as it was then known, at the Jet Propulsion Laboratory. This was also the group that developed the VICAR software. Consequently, the archetypal images used as models for the first IMAGE data objects were all in the VICAR file format.

The small bodies community, however, was developing a fondness for the transportability of FITS format, and so the Small Bodies Node saw PDS-labeled FITS images come to review in the first years of operation (mainly from ground-based observations like the coordinated observers of the International Halley Watch). Reviewers using the new NASAView tool to inspect the images reported a problem: the images as displayed by NASAView had their horizontal and vertical axes swapped, whereas software reading the FITS labels displayed the images correctly.

The reason for the discrepancy was easy enough to see when the FITS and VICAR format definitions were directly compared. Although both formats were considered "self-describing," that was only true to a point—the point at which something in the format definition itself constrained the data structure in the file. The FITS/VICAR label keywords described only the aspects of the file structure that were variable; their respective software support took care of the rest. The VICAR format description[12] specifies that a two-dimensional image (a 2D array in program memory) is stored in a VICAR file as a sequence of (horizontal) lines, or in row-major order as it was known to programmers of the day. The FITS standard, first defined by Wells et al. (1981),[13] specifies that a two-dimensional array ("image" being the most common application for a 2D array) is stored in a FITS file as a sequence of (vertical) columns, or in column-major order. Figure 1 illustrates the difference in storage order resulting from writing the elements of the same 2D array in memory first in row-major order and then in column-major order. The significance of file storage order only became apparent when third-party software, like NASAView, attempted to read the data files using only the information in the PDS3 labels and displayed results that were at odds with software that read the data using the native (VICAR or FITS) labels.

The first images archived with the PDS were VICAR images. There was no keyword to indicate storage order in the PSDD, and so the developers of NASAView interpreted the common storage order of the first image data sets as an inherent property of the PDS format and coded the display routine accordingly. This resulted in swapped axes for FITS images, a major question about images that might come in without either FITS or VICAR headers, and a significant problem for PDS image labels.

3.3. Further Complications

The solution to the storage order question would seem to be simple: add a keyword to the PSDD that indicates storage order and require all image objects to use it in their PDS3 labels. NASAView and other programs could then check that keyword's value and take the appropriate action. But display confusion resulted from more than just storage order, and adding a keyword was not as simple or effective a fix as it first appeared.

Although the FITS and VICAR file format specifications do indicate storage order, neither indicates which direction corresponds to "up." (Recall that the orientation of an image on a display device is the province of the software environment, not the file format specification). Figure 2 shows how the same storage order for pixels could result in three different display orientations, depending on the assumptions of the display software.

The PDS3 standard was also silent on the subject of image display orientation. In the case of PDS archival data, however, the display orientation could not be left as an exercise for the user, because PDS3 labels could, and often did, contain additional metadata beyond the simple fields required for identification and input/output. These metadata supported discovery and analysis and frequently included keywords for observational geometry that were defined in terms of angles related to "up" and "clockwise." The definitions of these essential keywords presupposed a correct image display orientation. A new keyword for storage order could sort out lines versus samples, but it still could not tell a display routine how to place the pixels correctly for interpretation.

3.4. The Solution

The solution applied was to define two new keywords for use in describing image objects in PDS3 labels: LINE_DISPLAY_DIRECTION and SAMPLE_DISPLAY_DIRECTION.[14] Both

[12] Available at http://www-mipl.jpl.nasa.gov/external/VICAR_file_fmt.pdf, accessed 2020 August 30.

[13] Current and historical versions available at https://fits.gsfc.nasa.gov/fits_standard.html.

[14] PDS3 syntax requires keywords to be in uppercase and their constituent terms separated by the underscore character ("_").

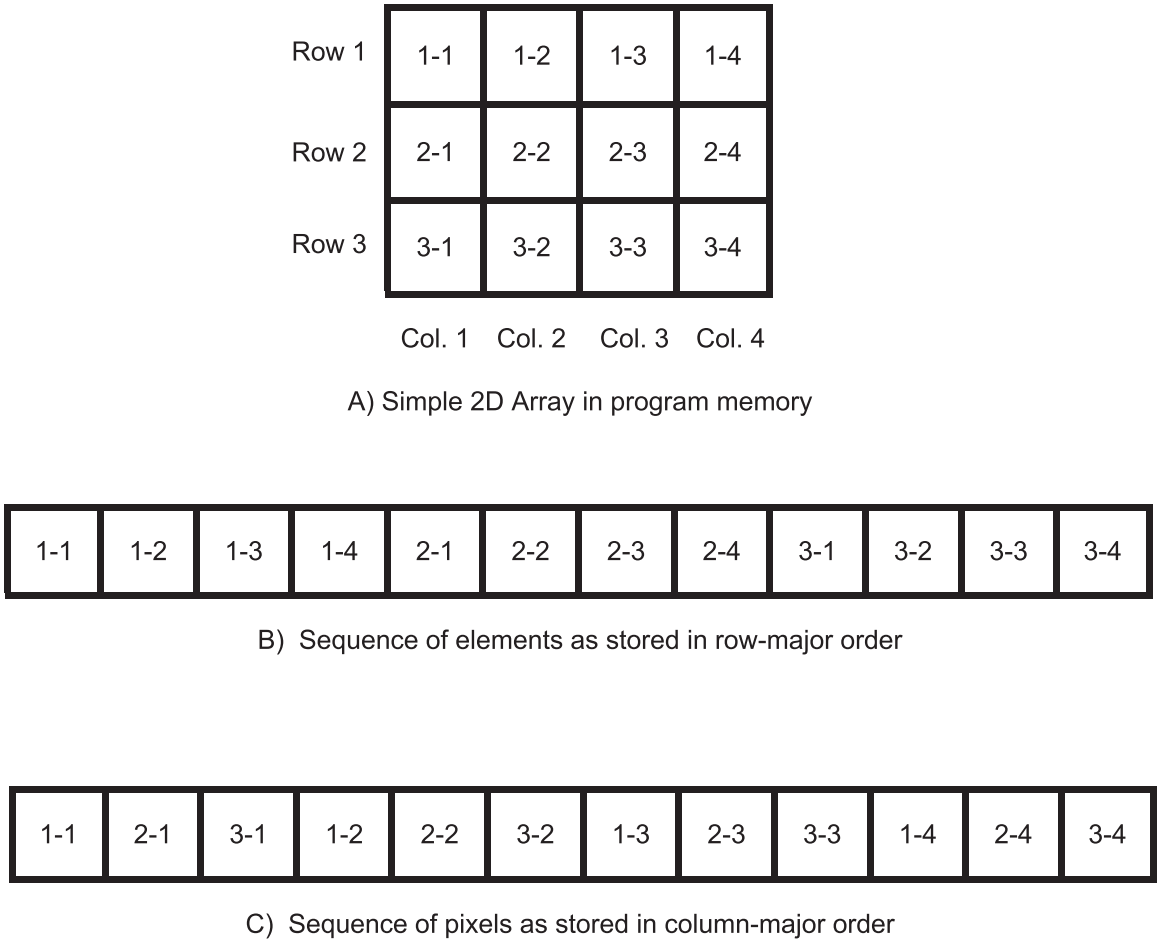


Figure 1. Element storage orders. (A) The conceptual diagram of a simple 2D array with its elements labeled by "row number—column number" as they might be organized in program memory. (B) The sequential order of the elements written out in the row-major storage order used by, e.g., the VICAR file format. (C) The sequential order of the same elements when the array is written out in the column-major order of the FITS file format.

keywords had the same set of four possible values, DOWN, UP, LEFT, or RIGHT, to indicate the direction in which successive pixels should be placed along the corresponding axis. So rather than indicating storage order, the sense of "line" and "sample" could be inverted for FITS images and the display routine directed to draw the "sample" dimension from the top DOWN, for example, rather than the more usual interpretation of left to RIGHT.

Confusing as this might have been for those coming from the FITS world, at least it would have solved the general display direction/storage order problem had it been consistently and universally applied across the PDS archive.

3.5. The Problem with the Solution

The solution, however, was neither consistently nor universally applied across the PDS, for two reasons.

First, the ODL version of the PSDD used by lvtool to validate labels had a relatively simple structure that supported only limited capabilities for validation enforcement. Inside a data object definition the only major validation constraint was one of requirement. That is, if a keyword was required to be present in, say, an IMAGE object and it was not found in the label, lvtool could flag the error. If the keyword was optional and not present, no flag was raised and the label passed validation. There was no version tracking in the system that would match keyword content in a PDS3 label to a particular version of the PDS3 standards; neither could lvtool infer whether there should be an additional keyword based on information included in the PSDD. Whether or not optional keywords were required in any particular set of labels was left to the nodes and the external peer reviewers.[15]

Second, the DISPLAY_DIRECTION keywords were not added as required elements because of a concern that legacy data would be "invalidated." Certainly, if the requirement were added and lvtool run on already-archived data, there would be an error flagged. This unlikely scenario was considered unacceptable. More practically, a programmer referencing the PSDD in order to understand potential label content would have been misled into assuming that all image data contained DISPLAY_DIRECTION keywords if they were listed in the PSDD as required. But legacy data, which of course were the bulk of the archive holdings at the time, would never have these keywords.

Why not add the DISPLAY_DIRECTION keywords to the legacy data if they were that important? The bulk of the legacy data sets were developed in the period 1987–1995, as part of the pilot project and early operations. Over that period of time CDROM supplanted magnetic tape as the long-term storage

[15] All data sets accepted for archiving in the PDS must pass an external peer review, convened by PDS.

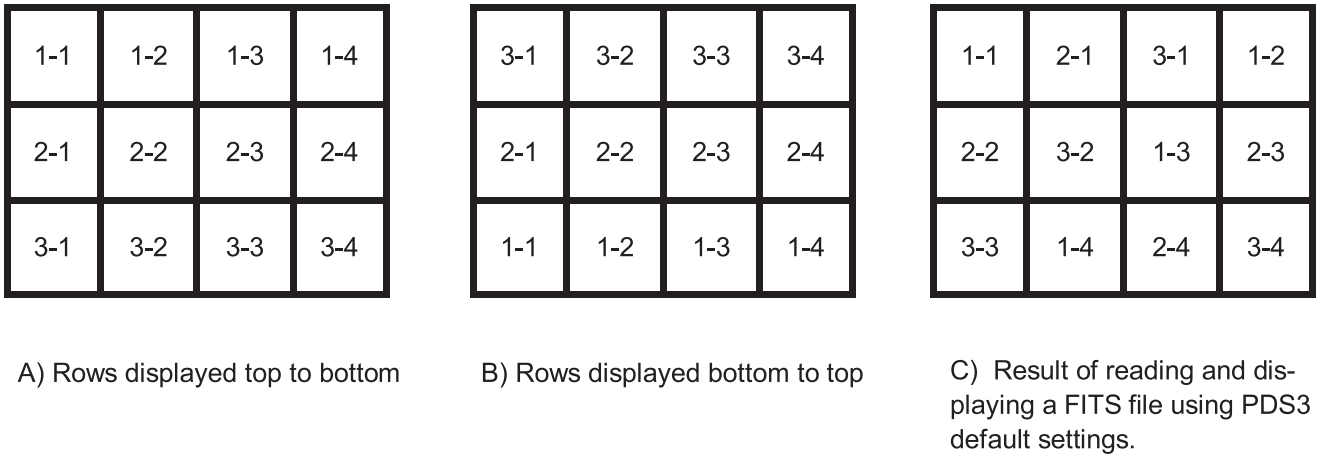


Figure 2. Effect of display settings. Once pixels are read from a file into memory, it is the purview of the software to arrange them on the display device. The same sequence of pixels can be displayed in various ways depending on assumptions about pixel order (or explicit user settings). (A) shows the he pixels from the previous example displayed in the PDS3 default display orientation, which places the first pixel in the upper left corner. (B) shows the same pixels, read in the same order, but displayed in the default orientation of IDL, which places the first pixel in the lower left corner. For comparison, panel (C) shows what would result if the original image had been stored in the column-major FITS format, but both read and displayed with the PDS3 default (row-major) settings.

medium of choice and the legacy data were, for the most part, on CDROM. But these were commercially produced CDROMs that were mastered and printed in runs of several hundred, then shipped to waiting users. This was an expensive and time-consuming process. The CDROM sets from a single mission contained as many as several hundred individual disks. PDS had no budget and no resources for updating labels for that volume of data, let alone reproducing the archive disks and redistributing them to the users who had received the original sets.

Rather than incurring the expense of remaking the legacy archives, the definitions of the DISPLAY_DIRECTION keywords included default values to be assumed in their absence. Unfortunately, not all data sets archived without DISPLAY_DIRECTION keywords, most notably the earlier FITS files, adhered to that default.

As time passed, image data came from more and more sources, non-VICAR and non-FITS, and this solution turned out to be just the first of many incorporated into the PSDD without any programmatically enforceable validation requirements. For all data structures, not just images, constraints that could not be enforced by lvtool were documented in the human-readable definitions contained in the PSDD and SR but relied on node personnel and reviewers for enforcement. Other significant problematic cases affecting data in various structures included geometric vectors that should have been accompanied by a set of coordinate system keywords; keywords that had different interpretations based on context, as documented in the PSDD textual descriptions; and keywords describing details of the QUBE data structure (primarily used for spectral image cubes) that users were directed (by the SR) to invent on the fly.

In short and in general, users had to consult the SR and the human-readable definitions in the PSDD to correctly interpret label content, and node personnel tasked with validating labels had an ever more arduous task of detailed manual validation as the years passed. Any programmer attempting to write generalized access routines found very little solid ground to work with.

3.6. Data Types

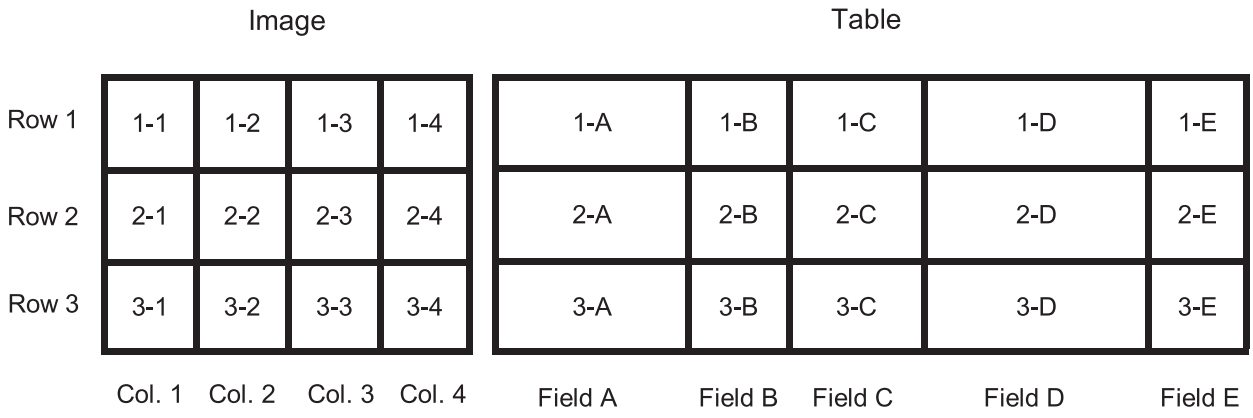
While pixel storage and display order were providing ample confusion for users, another aspect of technological evolution further complicated the process of simply reading the data.

When the first data structures where defined in the PSDD, the hardware numeric data types were identified by the terminology of the day, which tended to reference the manufacturer. The PSDD included data types like “MAC_REAL,” “SUN_REAL,” and “VAX_REAL.” But these data types were, more accurately, tied to the manufacturer of the chipset, rather than the chassis, of any given computer. As manufacturers diversified their chipset offerings, the original PSDD tags for data types led to confusion and mislabeling resulting from nameplate-based assumptions. The most typical discrepancy came from byte ordering, but in the case of the VAX formats there were some floating-point data types that were unique to the VAX architecture and had no IEEE equivalent.

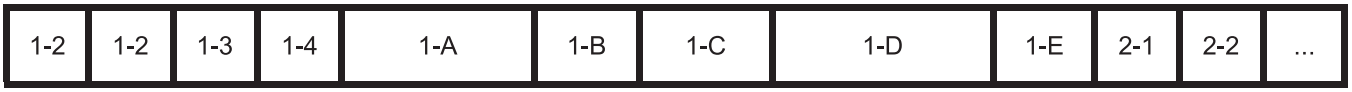
In the case of images, at least, it is relatively simple to recognize byte-order problems and, if necessary, determine the correct data type through brute-force attempts to read the data in all plausible formats. However, in more complex data structures, like binary tables, data types vary from field to field. Additional architectural considerations, like different byte orders for integers and real numbers, or the possibility of encountering BCD or EBCDIC fields, further complicate the processes of reading and displaying the data when these data types are encountered. As the data sets containing these data type identifications age, the ability to interpret them correctly becomes increasingly rare.

3.7. The Descriptive Approach

Finally, it is important to understand that for the first 10 yr of operations, PDS had essentially no enforcement authority. It could not require mission teams to format or reformat their data; it could advise, coax, and cajole, but it could not require. The worst penalty PDS could impose for noncompliance was to report to NASA that a team failed to meet its archiving requirement, and in those early years that had no practical consequence. This impacted the PDS3 standards.



A) Image and Table. The image pixels have been labeled as before. The table fields are identified by *row number-field letter.*



B) Interleaved data structures result in the stream of pixels being interrupted by rows of the table, and conversely.

Figure 3. Interleaved Data Objects. (A) The logical view of two data objects—and image and a table—stored in the same file. The image pixels are labeled as in previous figures; the table fields have been labeled using letters to distinguish them from pixels. PDS3 allowed tables like this to be described as "suffix bytes" of the image rows, and the image rows to be described as "prefix bytes" of the table rows. This resulted in the interleaved data structure shown in panel (B), where the byte stream coming from the data file must be decoded into rows of pixels alternating with rows of table fields. PDS4 explicitly prohibits such interleaving to minimize and localize the effect of misreading any one logical part of the data. An error in the length of table field A, for example, would not only corrupt the interpretation of the remainder of the table record but also lead to bytes from the table being taken for part of the image.

Recall that the PDS3 label language was intended to be able to describe any format that might be submitted for observational data. As more planetary missions were selected and funded, flight hardware became more sophisticated, as did the related processing pipelines. These pipelines organized their data files for efficiency within their local environments. When it came time to create the PDS archive submissions for their data, the archive developers looked at the PDS SR for the closest match to the pipeline data structure, and then the negotiation began.

Because the PDS3 labeling system was supposed to be descriptive (as opposed to prescriptive) and PDS had no authority to demand compliance with any specific format, the argument was made (and repeatedly won) that if something was not specifically prohibited by the SR, then it must be allowable. If a new accommodation in the SR was requested and it was comparable in any sense to something that was already allowed, then the SR should be modified, if needed, to include the new variation.

For example, the PDS3 IMAGE object, used to describe image arrays of two or three dimensions, had an option for each line to contain "prefix bytes" (or "suffix bytes," depending on the location) not considered part of the image. When in some cases these prefix/suffix bytes contained archival information, they needed to be described by another data structure, like a TABLE object (a fixed-width character or binary table format). The standards were expanded to allow prefix/suffix bytes on the ends of each row of a table. This led to file formats in which, for example, the left half of a file record could contain a scan line of an image and the right half could contain a row of a table (a structure known as "interleaved data objects," illustrated in Figure 3).

In another case, a mission produced simple two-dimensional images but decided to deliver them to PDS not as IMAGE objects but as degenerate QUBE objects. The QUBE object described the file structure used by the Integrated Software for Imagers and Spectrometers (ISIS) package for processing spectral image cubes with backplanes and sideplanes. The central cube and additional planes were interleaved in the data file. The choice of data object made these simple images inaccessible to non-ISIS users, because the QUBE object in the PDS3 label was not recognizable to their existing software.

In perhaps the most notable case, the Mars Global Surveyor (MGS) Thermal Emission Spectrometer (TES) instrument and the Cassini Composite Infrared Spectrometer (CIRS) instrument pipelines both produced data using a software package called Vanilla. The file format output by Vanilla included heap storage—an unstructured block of bytes accessed via a look-up table as an unordered set of variable-length records. This space-efficient design was at least partly in response to what was anticipated as being an overwhelming quantity of data to be returned—data volumes on the order of 100 TB at a time when data were stored and distributed on CDROM (~450–700 MB per disk) and DVD (4–8 GB per disk). The archive developers argued that the PDS3 standards contained a FILE object that did not require any structural description and therefore could be used to label the heap storage, and that the look-up table was a simple TABLE object. Therefore, the data format could be

described by the PDS3 label standards and should be accepted as archivable. And so, even though no PDS software, not even NASAView, could correctly interpret this label structure and display the data, the data sets were accepted in that format. The source code for Vanilla was included in the archive volumes, but software is not supported by PDS, which is not a software house.

Note also that both MGS and Cassini were archiving based on budgets set prior to the beginning of the operational PDS, so even if PDS had had the authority to enforce a format requirement, it is doubtful that either mission would have had the budget for a major reformatting effort.

3.8. Making the Standards Fit the Data

Even in cases where the PDS SR clearly did not support the intended format, data preparers successfully argued that it was the SR that should adapt. The most prominent example for this is compression. Frequently missions in planning expect to produce quantities of data that push the limits on contemporary physical storage. When this happens, the PDS node expecting to receive these data must find a way to handle the volume in the archive. Compression often seems like a good solution, but compression requires software support and software does not have archival lifetimes. Appendix I of the final, 3.8 version of the PDS Standards Reference describes two types of compression approved for use in the PDS3 archive: JPEG 2000 and ZIP. JPEG 2000 is an ISO standard that is not freely available; the URL for the Info-ZIP Consortium is no longer valid. If either of these formats ever becomes unsupported, PDS will need to rescue the data and migrate it to a new format.

4. Fewer, Simpler Formats

Throughout the history of PDS there has been a call from users for "fewer, simpler formats," but the perception of what constitutes data format is somewhat subjective. From the point of view of the PDS designers, a data format was what was defined by a standard. FITS was a format, VICAR was a format, and the PDS3 SR defined a format that was general enough to encompass both of those and many others as well. PDS users, however, had a different view.

4.1. The User View

Imagine a user wishing to compare PDS archived images from multiple data sources—some of which produced FITS images, some VICAR, and some PDS format with only the PDS3 label to document the image raster. In order to read all these images into the same analysis environment, this user is almost certainly going to have to write code to read at least one of these formats. If she chooses the PDS3 label as the common format, her code needs to account for the following variations:

1. Storage order as inferred from LINE_DISPLAY_DIRECTION and SAMPLE_DISPLAY_DIRECTION, if those keywords are present (checking for the presence of a FITS label in the data file if they are not).
2. The possible existence of prefix bytes on the beginning of each image scan line.
3. The possible existence of suffix bytes on the end of each image scan line.
4. The possibility of a third dimension indicated as "bands," and the three possible physical storage options for that third dimension.
5. The effect of the presence of "bands" on the storage of any prefix/suffix bytes on the image scan lines.
6. The hardware storage format for the pixels, specifically whether byte order is significant and, if so, if it is the same as the byte order of the computer reading the data.

This is complex logic to code just to read the data into memory.

The typical PDS data consumer is a researcher who writes code to support his own analysis and so tends to code to the data in hand rather than to a model. When that researcher obtains archive data "in PDS format" from a different data source and finds that his existing IMAGE-reading code cannot read the new data because of differences in the IMAGE data structure used in the new labels, he tends to view that as a difference in data format, irrespective of how the PDS designers viewed the issue.

Even the professional programmers hired to produce archive data were affected by the lack of PDS3 data structure constraints. Without tight constraints on data structures, each science pipeline modified existing structures to accommodate the design of the new science data center or instrument-specific data analysis and then expected the new design to become the archival format. Consequently, code developed for one archive production effort typically required substantial modifications for every new instrument or mission preparing an archive.

4.2. Whither OAL?

But what about the shareable libraries OAL and L3? They were provided to support users writing code to access PDS archive data, after all. Why would a user code "from scratch"?

This is another case where technological evolution overtook PDS development. Had PDS been operational even 5 yr earlier, it is likely that the OAL/L3 libraries would have been adopted by a fair number of users, who at that time were accustomed to working in coding environments that required compilation and linking. By the mid-1990s, however, users had several years' experience with analysis environments like the Astronomical Image Processing System (AIPS) and the Image Reduction and Analysis Facility (IRAF), as well as with scripting in the Interactive Data Language (IDL) and the Perl programming language. Users increasingly preferred the immediate results obtainable in those environments to the edit/compile/link process required for code written in C or FORTRAN. Consequently, the OAL and L3 libraries saw little use outside of the PDS-produced tools.

Had these libraries provided the primary access to the PDS3 data, as was intended, the end user's view of the PDS3 format would have been substantially different. The PDS programmers would have been managing the software changes required to support the evolution of the PDS3 Standards, and end users would have been insulated from the complexities of storage order, interleaving, and the like; "PDS3 format" would have been comparable to "FITS format" and "VICAR format" in practice. As it was, the users who had come to prefer the immediacy of scripting also had to deal personally with every modification made to the PDS3 Standards, and "PDS3 format" came to be viewed as an unpredictable conglomerate of structural possibilities.

5. PDS3, the Later Years

As PDS approached 15 yr of operations, it found itself firmly in the middle of not two but three opposing forces over issues of standards management and data formatting: first, there were the data preparers who were trying to meet budget and calendar deadlines and satisfy their NASA-imposed archiving requirements; second, there were the end users who wanted to be able to download and immediately proceed to analysis with any and all data they got from the PDS archives; and third, there were the future generations of users who had few if any contemporary advocates apart from the PDS itself. PDS was supposed to be preserving the data and maintaining their usability for those future generations, but in the early 2000s significant problems arose with certain legacy data sets that raised red flags for large sections of the archive.

5.1. Media Degradation

By about 2010, most PDS archival data were preserved via redundant copies on spinning disk, as opposed to CDROM, CDWO (CD-Write Once), or DVD. The Near-Earth Asteroid Rendezvous (NEAR) mission, however, ended in 2001, and the mission archives were written to DVD-R write-once media. One copy was shelved at SBN, one copy was deposited with the (then) National Space Science Data Center (NSSDC), and one copy was placed into a DVD jukebox at SBN for public access.

About a year later, users reported read failures attempting to read the NEAR data from the DVD jukebox. SBN investigated and found many read errors surfacing in all copies of these disks—the service copy and backup copy at SBN, and the deep-archive copy at the NSSDC. Other nodes were reporting similar issues, and a broader investigation within PDS determined that DVD-R was not a feasible archive medium because manufacturing standards were both generally low and unpredictable prior to purchasing the blank media. SBN eventually recovered the entire NEAR archive from the various DVD sets, but not without substantial effort and multiple attempts to read failing disks.

Contemporary reporting also called into question the claimed "100+ yr" shelf life for mastered CDs, and many of the CDs in the PDS archives were CDWO, which generally made more conservative claims for longevity. Clearly data on DVDs needed to be rescued immediately, and PDS needed to plan for media migration of data on CDs in the not-so-distant future.

5.2. Software-dependent Formats

The case of the Cassini CIRS data has already been mentioned as an example of warping the interpretation of the SR to fit the data format. Despite the size and importance of the Cassini CIRS (and MGS TES) data, the Vanilla software has not enjoyed wide community support. A version of the software was included on the archive disks along with a now 20 yr old "preliminary draft" of a user's guide, but ports were not forthcoming. The data were effectively inaccessible to anyone who did not have a working Vanilla installation, which was essentially everyone not on the original instrument team.

Additionally, the data from the Cassini Visual and Infrared Mapping Spectrometer (VIMS) instrument were archived in the QUBE format. This format presents many challenges for users attempting to write code to read these files: the central cube, sideplanes, and backplanes are all interleaved; the backplane dimensions (that is, record byte counts) do not match the cube dimensions; and because of the various interleaved data types, two-byte values do not always align with two-byte boundaries in memory. Consequently, the VIMS data set is all but unusable without the ISIS software.

In both cases, the PDS Ring-Moon Systems Node is spending significant node resources on reformatting these unique data sets into software-independent forms that can be read using more conventional techniques.

5.3. Archive Maintenance and Format Migration

As PDS looks to the future of research data, including assigning Digital Object Identifiers (DOIs) to data sets and enabling the programmatic access to discoverability and reusability that lie at the core of the FAIR Data Principles[16] (Wilkinson et al. 2016), the quantity and quality of metadata in the legacy archive decrease at least linearly with age, as does the surety with which the data can be incorporated into contemporary, interdisciplinary analysis. Media migration of data—copying data from old media to new, possibly different media—is an expense that can be managed as existing media age and new media are developed. Format migration—transforming the storage structure of the data, as is being done for the CIRS and VIMS data—is a far more resource-intensive process that frequently depends on living memory for the first- and second-generation PDS data sets. Data that are in orphaned formats now are at risk of being lost forever. The process of reformatting data is also potentially risky to the data, especially when the orphaned format includes hardware encodings that are no longer broadly supported, like VAX numerics or EBCDIC characters. These conversions require additional external peer review to verify that the reformatting has not degraded or damaged the data.

Knowing that the legacy PDS1, PDS2, and even PDS3 metadata may be incomplete—missing DISPLAY_DIRECTION, for example—adds further complexity and risk to an already costly process. The PDS3-to-PDS4 migration currently being undertaken by the PDS science nodes is a format migration, for example. For obsolete data types and data structures not supported under PDS4, format migration of the data files is required, with all the attendant validation, verification, and review. Even in cases where no format migration is required for the data files, the metadata in labels must undergo their own format migration from PDS3 ODL as defined by the PSDD to PDS4 XML as defined by the PDS4 Information Model. For older data the effort being spent to locate and document rich metadata to supplement what is in the original PDS3 labels in order to populate the PDS4 labels to the fullest extent possible represents a significant reinvestment in these legacy data.

Clearly, if an archive intends to preserve data for use by future generations, it is essential that the data be preserved in a way that maintains usability without requiring the periodic undertaking of risky (and expensive) format migration.

5.4. User Support

Even as PDS addresses the exigencies of preserving the legacy archive, it must also address the needs of contemporary users. Contemporary users expect to be able to search easily

[16] "FAIR" is an acronym for "Findable, Accessible, Interoperable, and Reusable," four principles to be applied in the management of research data.

across the entire PDS holdings, regardless of age, source, or physical location of the data. They want data in a format they can specify, so it can be immediately read into their preferred processing environment regardless of what format it might be in the archive. They expect the PDS holdings to be "discoverable" in the broad, contemporary sense of that term. The PDS holdings are now large enough, and the user community diverse enough, that it is no longer reasonable to assume that a PDS user knows what data are in the archive and what instrument or mission produced them.

Even if all the necessary metadata for that standard of support were in the PDS3 labels (they are not), the development of that level of service cannot be accommodated in a budget that also must account for repeated format migration of the aging archive. An archival format that precludes the need for regular format migration preserves resources for user services.

6. Characteristics of an Archival Data Format

As the first decade of the 2000s drew to a close, PDS determined that it needed to redesign its archive from the ground up, based on what it had learned in 20 yr of operations, in order to provide contemporary and future users with the national, interdisciplinary resource PDS was always intended to be.

It was clear that format migration would be needed in more than a few cases, to add missing metadata and to remove formats that had proved to be roadblocks to users. The PDS Management Council understood the risks involved and discussed the alternatives. If it had to be done, then certainly it should never be done twice. The PDS4 Data Design Working Group set to the task of defining a data format optimized for the long-term preservation of observational data.

The design team settled on the following characteristics:

1. Software Independence.
2. Simple Data Structures.
3. Contiguous Data Structures.
4. Rich Metadata.

6.1. Software Independence

This characteristic means there should be no assumptions about encoding, storage format, delimiters, etc. Everything should be explicitly stated in the label, regardless of how obvious it seems at the time of archiving. So, for example, offsets within a file currently must have an explicit unit of measure of "bytes" indicated, every time. Perhaps someday offsets will be measured in different units for some data structures ("characters" in Unicode files, for example), but if the unit is explicitly stated all the time now, there will be no need for defaults or assumptions documented elsewhere. Everything needed to read the data into memory will be in the label.

6.2. Simple Data Structures

In fact, there are four basic data structures in the PDS4 design, but only two are considered acceptable for observational data (the others are used for documentation and supplemental additions like browse images and thumbnails). These are arrays (multidimensional and homogeneous) and tables (repeating record structures). Any data in a more complex structure must be decomposed into a sequence of arrays and tables.

The reason for this goes straight to the heart of preservation. It is very difficult to misread a simple array or table, even if new code is written from scratch to do it. In the event of a coding error, the simplicity of the structures involved limits the types of errors it is possible to make, and those errors tend to leave a distinctive pattern in the data displayed. The same is true for labeling errors. If a keyword describing the structure makes it into the archive with an erroneous value, a standard PDS4 visualization tool will very likely show the characteristic error in its display. (Figure 4 illustrates some of the most commonly encountered data errors and their distinctive effects on image display.)

6.3. Contiguous Data Structures

Within the data file, the arrays and tables making up the data structures must be distinct. Interleaved formats like those created by prefix/suffix bytes on PDS3 TABLE and IMAGE objects are forbidden. Structures like the PDS3 QUBE object must be decomposed into the constituent parts, which may then be labeled as individual arrays and tables.

Requiring that the simple arrays and tables making up a complex data structure be separated into contiguous byte streams tends to limit the effect an error in one simple structure can have on reading the data in another. And again, the errors likely to be encountered have characteristic patterns on output. Misreading the start byte of an image array, for example, creates a "bar" pattern down one side of the image. Mislabeling the number of pixels in an image line creates a clear "wrapping" pattern (shown in Figure 4(D)). These patterns, as well as their potential sources, would be obscured if the image were interleaved with table elements.

6.4. Rich Metadata

When PDS was designed, traditional documentation was expected to supply the bulk of details regarding provenance, processing, and interpretation of the data. Now this information is expected to be in the metadata supplied in the label. Rich metadata support discovery, analysis, reuse, automated processing, and any other application of the data users might imagine. It can also support complex data transformations.

When a complex structure, like a PDS3 QUBE with backplanes and sideplanes, is deconstructed into its logical constituents, the associations between the logical pieces inherent in the original form are significant and should not be discarded. The PDS4 metadata system includes mechanisms for defining relationships between and among data structures to define and preserve these associations. So as new, complex data formats are developed, it is possible to deconstruct the data into their simple, logical components and define metadata for reconstructing the original format. The advantage here is twofold: first, a single program can be written to transform the native format to the archival equivalent, making the constituent simple data structures available to all applications without further transformation; and second, a single program can be written to transform any equivalent set of data structures to the new native format, irrespective of what source(s) produced the older data.

7. Trade-offs

As with any major system change, there are drawbacks and benefits.

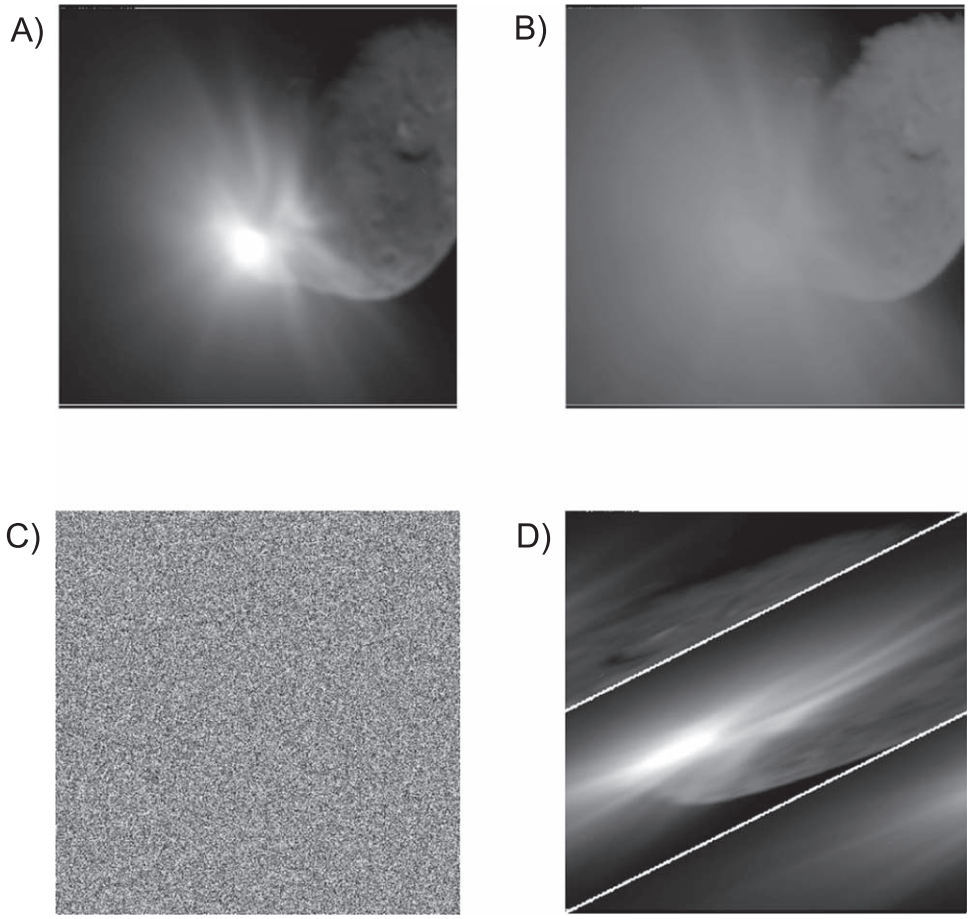


Figure 4. Effect of common read errors on a simple image. Panel (A) is a calibrated image from the Deep Impact mission High-Resolution Imager (McLaughlin et al. 2014), properly read and displayed. The pixels in this image are four-byte real numbers. Panel (B) shows the characteristic compression of dynamic range that results from misreading the same floating-point data as signed integer data. Panel (C) shows the telltale "snow" pattern that typically results from interpreting integer data in the wrong byte order. (Interpreting real data in the wrong byte order usually results in fatal errors because of bit patterns that coincide with NaN, +Inf, or −Inf values.) Finally, panel (d) shows the wrapping pattern that results when axes of unequal length are swapped or, as in this case, when one axis length is off by a small number of bytes. (The original square image has a row of bright pixels very near the top border that produces the sharp diagonal lines in panel (D).) All four images were produced by the PDS4_viewer from the same FITS data file, using rudimentary PDS4 labels with the specific errors introduced prior to reading the data and applying a z scale to the display.

There are two major drawbacks for PDS4 label approach compared to PDS3:

1. Data creators are now required to fit their data to the PDS4 data structures. This is a stark difference from the PDS3 approach, but PDS4 was designed to be prescriptive rather than descriptive specifically to get some measure of control over "data formats," as perceived by users.
2. The PDS4 archive format will require software support. In general, users will want data in a format other than the format that is in the archive. To date, the PDS-supplied transformation tool is severely limited in what it can do. As analytical environments become more sophisticated and users seek to integrate data from various sources, this demand is likely to increase.

The benefits to PDS and its present and future users, however, are great:

1. Because the PDS format constraints are tight and the data structures are explicitly defined in the PDS4 Information Model, programmers can code to the model for basic access to the data. This is most important for general read/write routines, of course. Unlike PDS3, it is possible to write code to do basic input and output on all array-type objects using the information that is required to be present in the PDS4 labels, for example, regardless of what instrument or mission produced the data.
2. Generic data structures that are not tied to any specific processing environment are essential to supporting broad investigations. A user should not have to understand the differences between FITS, VICAR, and PDS3 IMAGE data storage structures, for example, before being able to compare images.
3. If PDS does not have to allocate resources to format migration, it can redirect those resources to user services. An archive that needs constant maintenance and migration to remain viable will require ever-increasing resources as the archive grows. A stable archive format may not be the preferred format of contemporary users, but the preferred format is only a transformation away. Writing one transformation that will work on the entire archive is far more cost-effective than repeated format migration of archive content.

8. Summary

The PDS was designed in an age where data were exchanged on magnetic tape, users routinely programmed in compiled languages like FORTRAN, and the greatest challenge users faced was being able to find and lay hands on the data. The PDS3 standards for labeling and formatting data sought to emulate the success of the FITS and VICAR formats but were overwhelmed by the technological evolution that saw online access replace physical distribution, interactive environments replace compiled code, and Google supplanting traditional card-catalog searching.

The PDS3 standards were a descriptive approach to labeling data that aimed to provide the flexibility to document any data file structure provided, but they were themselves overtaken by the complexity and variety of data organizations used by newer and more sophisticated instruments. Ultimately, in order to provide users with the core services now considered essential in an archive, the PDS needed a data file format optimized for archiving; a format that would provide a sound basis for programmatic access, use, and reuse; and a format that would be usable without change for generations, so that PDS resources can be focused on user services rather than expensive and risky maintenance operations on legacy data. The PDS4 data format is designed to meet those criteria.

This work was supported in part by the NASA PDS Small Bodies Node through NASA grant NNX16AB16A. The authors would like to thank Mark Showalter and Mitch Gordon from the NASA PDS Ring-Moon Systems Node for their assistance and clarifications regarding the ongoing Cassini VIMS and CIRS work.

The original data product and the modified version with accompanying PDS4 label mock-ups used to create the images in Figure 4 are available at doi:10.5281/zenodo.5171393.

ORCID iDs

Anne Raugh https://orcid.org/0000-0002-8300-9443
J. Steven Hughes https://orcid.org/0000-0003-4851-293X

References

National Research Council 1982, Data Management and Computation Volume 1: Issues and Recommendations (Washington, DC: The National Academy Press), doi:10.17226/12366
National Research Council 1986, Issues and Recommendations Associated with Distributed Computation and Data Management Systems for the Space Sciences (Washington, DC: The National Academies Press), doi:10.17226/12343
Kieffer, H., Arvidson, R. E., Baum, W. A., et al. 1984, Planetary Data Workshop, NASA Conf. Pub. 2343, Part 1 (Washington, DC: NASA), 2343, https://ntrs.nasa.gov/citations/19840026295
McLaughlin, S. A., Carcich, B., Sackett, S. E., et al. 2014, Deep Impact 9P/Tempel Encounter—Reduced HRIV Images V3.0, DIF-C-HRIV-3/4-9P-ENCOUNTER-V3.0, https://pdssbn.astro.umd.edu/holdings/dif-c-mri-3_4-9p-encounter-v3.0/
Planetary Data System 2021, Planetary Data System Standards Reference, Version 1.16.0, JPL D-7669 (Pasadena, CA: JPL, Caltech), https://pds.nasa.gov/datastandards/documents/sr/v1/StdRef_1.16.0.pdf
Planetary Data System Design Team (PDS-DT) 1986, Planetary Data System Version 1.0 Specification, Revision 1.0, JPL Document D-3454, doi:10.17189/1519398
Renfrow, J. T., Martin, M. D., Jansma, P. A., et al. 1986, Planetary Data System System Requirements Review, doi:10.17189/1519402
Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363
Wilkinson, M., Dumontier, M., Aalbersberg, I., et al. 2016, Scientific Data, 3, 160018