JPL D-7669, Part 2

Planetary Data System Standards Reference

February 27, 2009
Version 3.8

PDS
Planetary Data System
National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

(This page intentionally left blank.)

PDS Standards Reference
Table of Contents

Chapter 1. Introduction 1-1
1.1 PDS Data Policy 1-1
1.2 Purpose 1-1
1.3 Scope 1-1
1.4 Audience 1-1
1.5 Document Organization 1-2
1.6 Other Reference Documents 1-2
1.7 Online Document Availability 1-3

Chapter 2. Cartographic Standards 2-1
2.1 Introduction 2-1
2.1.1 International and NASA Advisory Groups for Cartographic Stds 2-1
2.2 Inertial Reference Frame and Time System 2-2
2.3 Spin Axes and Prime Meridians 2-3
2.4 Body-Fixed Planetary Coordinate Systems 2-4
2.4.1 Planets and Satellites 2-5
2.4.2 Small Bodies 2-6
2.4.3 Rings 2-6
2.4.4 Planetary Plasma Interactions 2-6
2.5 Surface Models 2-7
2.6 PDS Keywords for Cartographic Coordinates 2-7
2.7 Map Resolution 2-9
2.8 References 2-10

Chapter 3. DATA_TYPE Values and Data File Storage Formats 3-1
3.1 Data Elements 3-1
3.2 Data Types 3-1
3.3 Binary Integers 3-4
3.4 Signed vs. Unsigned Integers 3-4
3.5 Floating Point Formats 3-5
3.6 Bit String Data 3-5
3.7 Character Data 3-5
3.8 Format Specifications 3-5
3.9 Internal Representations of Data Types 3-6

Chapter 4. Data Objects and Products 4-1
4.1 Data Product File Configurations 4-2

Chapter 5. Data Product Labels 5-1
5.1 Format of PDS Labels 5-1
5.1.1 Labeling methods 5-1
5.1.2 Label format 5-1

5.2 Data Product Label Content 5-4
5.2.1 Attached and Detached Labels 5-4
5.2.2 Combined Detached Labels 5-4
5.2.3 Minimal Labels 5-6
5.2.3.1 Implicit File Object (Attached and Detached Minimal Label) 5-8
5.2.3.2 Explicit File Object (Detached Minimal Label) 5-8
5.3 Detailed Label Contents Description 5-8
5.3.1 Label Standards Identifiers 5-9
5.3.2 File Characteristic Data Elements 5-10
5.3.3 Data Object Pointers 5-11
5.3.3.1 Use of Pointers in Attached Labels 5-11
5.3.3.2 Use of Pointers in Detached and Combined Detached Labels 5-12
5.3.3.3 Note Concerning Minimal Attached and Detached Labels 5-14
5.3.4 Data Identification Elements 5-14
5.3.4.1 Spacecraft Science Data Products 5-15
5.3.4.2 Earthbased Science Data Products 5-15
5.3.4.3 Ancillary Data Products 5-15
5.3.5 Descriptive Data Elements 5-16
5.3.6 Data Object Definitions 5-16
5.3.7 End Statement 5-17
5.4 Syntax for Element Values 5-17
5.5 Locally-defined Data Elements 5-18
5.5.1 Justification for Locally-defined Data Elements 5-18
5.5.2 Identification of Locally-defined Data Elements 5-20
5.5.3 Review and Use of Locally-defined Data Elements 5-20

Chapter 6. Data Set/Data Set Collection Contents and Naming 6-1
6.1 Data Set Naming and Identification 6-2
6.2 Data Set Collection Naming and Identification 6-3
6.3 Name and ID Components 6-4
6.3.1 Restrictions on DATA_SET_ID and DATA_SET_COLLECTION_ID 6-4
6.3.2 Standard Acronyms, Abbreviations, and Assignments 6-4
6.4 Examples 6-8

Chapter 7. Date/Time Format 7-1
7.1 Date/Times 7-1
7.2 Dates 7-2
7.2.1 Conventional Dates 7-2
7.2.2 Native Dates 7-2
7.3 Times 7-2
7.3.1 Conventional Times 7-2
7.3.2 Native Times 7-3
7.4 Midnight and Leap Seconds 7-3
7.4.1 Midnight 7-4
7.4.2 Leap Seconds 7-4

Chapter 8. Directory Types and Naming ... 8-1
8.1 Standard Directory Names ... 8-1
8.2 Formation of Directory Names ... 8-2
8.3 Path Formation Standard ... 8-4
8.4 Tape Volumes ... 8-4
8.5 Exceptions to These Standards ... 8-4

Chapter 9. Documents ... 9-1
9.1 PDS Objects for Documents ... 9-2
9.1.1 TEXT Objects ... 9-2
9.1.2 DOCUMENT Objects ... 9-2
9.2 Document Format Details ... 9-3
9.2.1 Flat ASCII Text ... 9-3
9.2.2 ASCII Text Containing Markup Language ... 9-4
9.2.2.1 Hyper-Text Markup Language (HTML) Files ... 9-4
9.2.2.2 Location of Files ... 9-4
9.2.2.3 Discouraged HTML 3.2 Capabilities ... 9-4
9.2.3 Non-ASCII Formats ... 9-5
9.2.4 Validation ... 9-5
9.3 Examples ... 9-5
9.3.1 Simple Example of Attached label (Plain ASCII Text) ... 9-5
9.3.2 Complex Example of Detached Label (Two Document Versions) ... 9-5
9.3.3 Complex Example of Detached Label (Documents Plus Graphics) ... 9-6

Chapter 10. File Specification and Naming ... 10-1
10.1 File Specification Standards ... 10-1
10.1.1 ISO 9660 Level 1 Specification ... 10-2
10.1.2 ISO 9660 Level 2 Specification ... 10-2
10.1.3 Specification for Files Delivered Electronically ... 10-3
10.2 Reserved Directory Names, File Names and Extensions ... 10-3
10.2.1 Reserved Directory Names ... 10-3
10.2.2 Reserved File Names ... 10-3
10.2.3 Reserved Extensions ... 10-4
10.3 Guidelines for Naming Sequential Files ... 10-4

Chapter 11. Media Formats for Data Submission and Archive ... 11-1
11.1 CD-ROM Recommendations ... 11-2
11.1.1 Use of Variant Formats ... 11-2
11.1.2 Premastering Recommendation ... 11-2
11.2 DVD Recommendations ... 11-2
11.2.1 Use of Variant Formats ... 11-2
11.2.2 Premastering Recommendation ... 11-2
11.2.3 Recommended DVD Formats ... 11-2
11.3 Packaging Software Files on a CD or DVD ... 11-3
11.4 Software Packaging Under Previous Versions of the Standard ... 11-3

Chapter 12. Object Description Language Specification and Usage.................. 12-1
12.1 About the ODL Specification 12-1
12.1.1 Implementing ODL 12-2
12.1.1.1 Language Subsets 12-2
12.1.1.2 Language Supersets 12-2
12.1.1.3 PDS Implementation of PVL-Specific Extensions 12-2
12.1.2 Notation 12-3
12.2 Character Set 12-3
12.2.1 ODL Character Set - Letters 12-4
12.2.2 ODL Character Set - Digits 12-4
12.2.3 ODL Character Set - Special Characters 12-4
12.2.4 ODL Character Set - Spacing Characters 12-5
12.2.5 ODL Character Set - Format Effectors 12-5
12.2.6 ODL Character Set - Control Characters 12-6
12.3 Lexical Elements 12-6
12.3.1 Numbers 12-6
12.3.1.1 Integer Numbers in Decimal Notation 12-6
12.3.1.2 Integer Numbers in Based Notation 12-7
12.3.1.3 Real Numbers 12-7
12.3.2 Dates and Times 12-8
12.3.2.1 Date and Time Values 12-8
12.3.2.2 Implementation of Dates and Times 12-9
12.3.2.3 PDS Implementation of Dates and Times 12-9
12.3.2.4 Dates 12-9
12.3.2.5 Times 12-9
12.3.2.5.1 Combining Date and Time 12-10
12.3.3 Strings 12-10
12.3.3.1 Text Strings 12-10
12.3.3.2 Symbol Strings 12-11
12.3.4 Identifiers 12-11
12.3.4.1 Reserved Identifiers 12-12
12.3.5 Special Characters 12-12
12.4 Statements 12-12
12.4.1 Lines and Records 12-13
12.4.2 Attribute Assignment Statement 12-14
12.4.3 Pointer Statement 12-14
12.4.4 OBJECT Statement 12-15
12.4.4.1 Implementation of OBJECT Statements 12-16
12.4.5 GROUP Statement 12-16
12.4.5.1 Implementation of GROUP Statements 12-17
12.4.5.2 PDS Usage of GROUP 12-17
12.5 Values 12-17
12.5.1 Numeric Values 12-17
12.5.2 Units Expressions 12-18
12.5.2.1 Implementation of Numeric Values 12-18
12.5.3 Text String Values 12-19

12.5.3.1 Implementation of String Values 12-19
12.5.3.1.1 PDS Text String Formatting Conventions 12-20
12.5.4 Symbolic Literal Values 12-20
12.5.4.1 Implementation of Symbolic Literal Values 12-21
12.5.4.2 PDS Convention for Symbolic Literal Values 12-21
12.5.5 Sequences 12-21
12.5.6 Sets 12-22
12.5.6.1 PDS Restrictions on Sets 12-22
12.6 ODL Summary 12-22
12.7 Differences Between ODL Versions 12-24
12.7.1 Differences from ODL Version 1 12-24
12.7.1.1 Ranges 12-24
12.7.1.1.1 Delimiters in Sequences and Sets 12-25
12.7.1.1.2 Exponentiation Operator in Units Expressions 12-25
12.7.2 Differences from ODL Version 0 12-25
12.7.2.1 Date-Time Format 12-25
12.7.3 ODL/PVL Usage 12-25

Chapter 13. PDS Objects / Groups 13-1
13.1 Generic and Specific Data Object Definitions 13-1
13.1.1 Primitive Objects 13-2
13.2 Generic and Specific Data Group Definitions 13-3
13.2.1 Implementation of Group Statements 13-4

Chapter 14. Pointer Usage 14-1
14.1 Types of Pointers 14-1
14.1.1 Data Location Pointers (Data Object Pointers) 14-1
14.1.2 Include Pointers 14-1
14.1.3 Related Information Pointers (Description Pointers) 14-2
14.2 Rules for Resolving Pointers 14-2

Chapter 15. Record Formats 15-1
15.1 FIXED_LENGTH Records 15-1
15.2 STREAM Records 15-2
15.3 VARIABLE_LENGTH Records 15-2
15.4 UNDEFINED Records 15-3

Chapter 16. SFDU Usage 16-1
16.1 The ZI SFDU Organization 16-2
16.2 The ZKI SFDU Organization 16-5
16.3 Examples 16-7
16.4 Exceptions to this Standard 16-8

Chapter 17. Usage of N/A, UNK and NULL 17-1
17.1 Interpretation of N/A, UNK, and NULL 17-1

17.1.1 N/A ... 17-1
17.1.2 UNK ... 17-1
17.1.3 NULL ... 17-1
17.2 Implementation Recommendations for N/A, UNK, and NULL ... 17-2

Chapter 18. Units of Measurement ... 18-1
18.1 SI Units ... 18-1

Chapter 19. Volume Organization and Naming ... 19-1
19.1 Volume Set Types ... 19-1
19.2 Volume Organization Guidelines ... 19-2
19.3 Description of Directory Contents and Organization ... 19-2
19.3.1 ROOT Directory Files ... 19-2
19.3.2 Required Subdirectories ... 19-8
19.3.2.1 CATALOG Subdirectory ... 19-8
19.3.2.2 Data Subdirectory ... 19-9
19.3.2.3 INDEX Subdirectory ... 19-10
19.3.3 Optional Subdirectories ... 19-11
19.3.3.1 CALIBration Subdirectory ... 19-11
19.3.3.2 DOCUMENT Subdirectory ... 19-12
19.3.3.3 EXTRAS Subdirectory ... 19-13
19.3.3.4 GAZETTER Subdirectory ... 19-13
19.3.3.5 GEOMETRY Subdirectory ... 19-14
19.3.3.6 LABEL Subdirectory ... 19-14
19.3.3.7 SOFTWARE Subdirectory ... 19-14
19.4 Volume Naming ... 19-16
19.4.1 Volume ID ... 19-16
19.5 Volume Set Naming ... 19-17
19.5.1 Volume Set ID ... 19-17
19.6 Logical Volume Naming ... 19-18
19.7 Exceptions to This Standard ... 19-18

Appendix A. PDS Data Object Definitions ... A-1
A.1 ALIAS ... A-3
A.2 ARRAY (Primitive Data Object) ... A-4
A.3 BIT_COLUMN ... A-8
A.4 BIT ELEMENT (Primitive Data Object) ... A-11
A.5 CATALOG ... A-12
A.6 COLLECTION (Primitive Data Object) ... A-15
A.7 COLUMN ... A-16
A.8 CONTAINER ... A-20
A.9 DATA_PRODUCER ... A-27
A.10 DATA_SUPPLIER ... A-29
A.11 DIRECTORY ... A-31
A.12 DOCUMENT ... A-33
A.13 ELEMENT (Primitive Data Object) ... A-36

A.14 FIELD A-38
A.15 FILE A-41
A.16 GAZETTEER_TABLE A-45
A.17 HEADER A-55
A.18 HISTOGRAM A-57
A.19 HISTORY A-60
A.20 IMAGE A-64
A.21 INDEX_TABLE A-69
A.22 PALETTE A-74
A.23 QUBE A-77
A.24 SERIES A-85
A.25 SPECTRAL_QUBE A-90
A.26 SPECTRUM A-107
A.27 SPICE KERNEL A-110
A.28 SPREADSHEET A-113
A.29 TABLE A-118
A.30 TEXT A-139
A.31 VOLUME A-141
A.32 WINDOW A-148

Appendix B. Complete PDS Catalog Object Set B-1
B.1 DATA_SET B-4
B.2 DATA_SET_COLL_ASSOC_DATA_SETS B-11
B.3 DATA_SET_COLLECTION_REF_INFO B-12
B.4 DATA_SET_COLLECTION B-13
B.5 DATA_SET_COLLECTION_INFO B-16
B.6 DATA_SET_HOST B-18
B.7 DATA_SET_INFORMATION B-19
B.8 DATA_SET_MAP_PROJECTION B-22
B.9 DATA_SET_MAP_PROJECTION_INFO B-25
B.10 DATA_SET_MISSION B-27
B.11 DATA_SET_REFERENCE_INFORMATION B-28
B.12 DATA_SET_TARGET B-29
B.13 DS_MAP_PROJECTION_REF_INFO B-30
B.14 IMAGE_MAP_PROJECTION B-31
B.15 INSTRUMENT B-36
B.16 INSTRUMENT_HOST B-41
B.17 INSTRUMENT_HOST_INFORMATION B-43
B.18 INSTRUMENT_HOST_REFERENCE_INFO B-44
B.19 INSTRUMENT_INFORMATION B-45
B.20 INSTRUMENT_REFERENCE_INFO B-48
B.21 INVENTORY B-49
B.22 INVENTORY_DATA_SET_INFO B-51
B.23 INVENTORY_NODE_MEDIA_INFO B-52
B.24 MISSION B-53
B.25 MISSION_HOST B-59

B.26 MISSION_INFORMATION B-60
B.27 MISSION_REFERENCE_INFORMATION B-62
B.28 MISSION_TARGET B-63
B.29 PERSONNEL B-64
B.30 PERSONNEL_ELECTRONIC_MAIL B-66
B.31 PERSONNEL_INFORMATION B-67
B.32 REFERENCE....B-68
B.33 SOFTWARE B-75
B.34 SOFTWARE_INFORMATION B-77
B.35 SOFTWARE_ONLINE B-78
B.36 SOFTWARE_PURPOSE B-79
B.37 TARGET....B-80
B.38 TARGET_INFORMATION....B-82
B.39 TARGET_REFERENCE_INFORMATION....B-83

Appendix C. Internal Representation of Data Types C-1
C.1 MSB_INTEGER C-2
C.2 MSB_UNSIGNED_INTEGER....C-4
C.3 LSB_INTEGER C-6
C.4 LSB_UNSIGNED_INTEGER....C-8
C.5 IEEE_REAL C-10
C.6 IEEE_COMPLEX....C-13
C.7 PC_REAL C-14
C.8 PC_COMPLEX C-17
C.9 VAX_REAL, VAXG_REAL C-18
C.10 VAX_COMPLEX, VAXG_COMPLEX C-22
C.11 MSB_BIT_STRING C-23
C.12 LSB_BIT_STRING C-25

Appendix D. Examples of Required Files.... D-1
D.1 AAREADME.TXT D-2
D.2 INDXINFO.TXT D-8
D.3 SOFTINFO.TXT.... D-9
D.4 VOLDESC.CAT D-13

Appendix E. NAIF Toolkit Directory Structure E-1
E.1 NAIF Directory....E-2
E.2 TOOLKIT Directory....E-3
E.3 Using the NAIF Toolkit....E-12
E.4 NAIF's File Naming Conventions....E-13

Appendix F. Acronyms F-1

Appendix G. SAVED Data G-1

G.1 Safekeeping Process and Procedures G-1
G.2 Safekeeping Standards G-1

Appendix H. PDS Data Group Definitions H-1
H.1 BAND_BIN H-3
H.2 BAND_SUFFIX H-4
H.3 LINE_SUFFIX H-5
H.4 PARAMETERS H-6
H.5 SAMPLE_SUFFIX H-7

Appendix I. Data Compression Formats I-1
I.1 CLEM-JPEG I-3
I.2 HUFFMAN FIRST DIFFERENCE I-4
I.3 JPEG 2000 ..... I-5
I.4 PREVIOUS PIXEL I-10
I.5 RUN LENGTH I-11
I.6 ZIP I-12

(This page intentionally left blank.)

PDS Standards Reference Change Log

Version	Section	Change
3.1	1.1	PDS Data Policy added
	2.3	Reference coordinate standard expanded to support body-fixed rotating, body-fixed non-rotating, and inertial coordinate systems.
	2.4	Ring coordinate standard added.
	3.0	List of internal representations of data types moved to Appendix C
	3.2	EBCDIC_CHARACTER added to PDS Standard data types
	5.2.3	Minimal label option described
	6.3	Data set collection naming -- data processing level component made optional
	6.4	Data set naming -- added support for SPICE and Engineering, where no instrument component applies
	10.0, ALL	PDS use of UNIX/POSIX forward slash separator for path names. VMS-style bracket notation replaced.
	10.2.1	Required file names for catalog objects included
	12.5.4.2	PDS use of double quotes clarified
	13.2	Use of Primitive objects described
	14	New chapter -- Pointer Usage
	17	New chapter -- PDS Usage of N/A, UNK, and NULL
	19	Logical Volume organization added
	Appendix A	Primitive Objects added
	Appendix A	Header object -- required and optional keyword lists changed Container object -- Column no longer a requried sub-object

Version	Section	Change
	Appendix B	Streamlined Catalog Object Templates with examples replace 3.0 set
	Appendix C	New appendix containing internal representations of data types (moved from Chapter 3)
	Appendix D	Outline and example for AAREADME.TXT added
	Appendix E	Version 3.0 Acronyms and Abbreviations modified and moved to this Appendix. Spelling and Word Usage section deleted.
	Index	The document now features an index.
	ALL	No other substantive changes have been made to the standards since the release of Version 3.0. Throughout the document, clarifications have been made, typos corrected, some sections have been rearranged, and new examples have been supplied.

Version	**Section**	**Change**
3.2		Release Date: 7/24/95
	5.1.2	Label format discussion added Noted that values in labels should be upper case (except descriptions). Fixed examples in Appendix A.
	5.2.3, Appendix A	Noted that for data products using minimal labels, DATA_OBJECT_TYPE = FILE in the Data Set Catalog Template
	6	Added target IDs for DUST and SKY Added instrument component values SEDR and POS Noted that Data Set and Data Set Collection IDs and Names should be upper case. Fixed examples.
	8 and 19	Listed CALIB and GEOMETRY as recommended directory names (as opposed to required).
	8.2	SOFTWARE Subdirectory naming recommendation added
	9.1	Volumes may contain multiple versions of VOLINFO

9.2.1	Increased maximum line length in text file to 78 characters plus CR/LF
10.1	Clarified file name spcification. Noted that file name must be upper case and that full stop character required
10.2	Added recommendation that file extension identify the data type of a file. Added .QUB as reserved file extension for spectral image qubes. Added SPICE file extensions to reserved file extension list. catalog pointer name and file name: SWINV.CAT Added LABINFO.TXT to list of required xxxINFO.TXT files. Added recommended xxx INFO.TXT file names for SOFTWARE subdirectories.
10.2.3 and 5.1	added note that detached label file (*.LBL) should have the same base name as the associated data file
11.1.1	Added PDS Extended Attribute Record (XAR) policy
11.1.2	Added recommendation that CDs be premastered using single-session, single-track format.
11.1.3	Added section on Packaging Software files on a CD-ROM
14.1.2	Added new example of structure pointer
15	Added recommendation that for VAX/VMS-compatible CDs, fixed length and variable length files be an even number of bytes. Removed reference to VMS restriction to an even number of bytes in section 15.2
15.1	Removed discussion of use of BLOCK_BYTES and BLOCKING_TYPE (since this data element not in PSDD)
15.3	Added notation that CR/LF is required line terminator for PDS label and catalog files
15.5	Reworded first sentence.
17.2	Allow definition of numeric constants representing N/A, UNK, and NULL to be defined for use in an INDEX table.
18	replaced reference to PDS V1.0 with a general statement

19	Added SOFTWARE subdirectory recommendations
19	Recommend that an archive volume be based on a single version of the PDS standards. Volume organization guidelines added.
19.2	Clarified requirements for files & directories when logical volumes used
19.3	INDEX table standard update
19.3	use of axx- and bxx- prefixes in required file names clarified
19.4, Appendix A	fixed examples--Volume and Volume set names capitalized
19.5.1	Volume set ID formation rule modified.
Appendix A	updated COLUMN, BIT_COLUMN, and HISTOGRAM objects required and optional keyword lists to be consistent with Table 3.1
Appendix A	Added ALIAS and INDEX_TABLE objects
Appendix A	Added examples of COLUMN objects having ITEMs
Appendix A	Clarified use of ROW_SUFFIX_BYTES and ROW_PREFIX_BYTES for SPARE fields in Tables with fixed length records
Appendix A	Clarified the requirements for VOLUME objects for Logical volumes
Appendix A	Fixed examples using HEADER object to conform to current standard. Modified description of Header object to eliminate confusion..
Appendix B	Inventory, Software_Inventory and Target templates added
Appendix B	Removed incorrect example of use of Personnel template
Appendix D	INDXINFO.TXT and SOFTINFO.TXT outlines and examples added
Appendix D.1	Modified example of AAREADME.TXT to include rules on how pointer statements are resolved.

	Appendix E and F	Added Appendix E - NAIF Toolkit Directory Structure. Acronyms and Abbreviations moved to Appendix F.
	ALL	corrected typos, clarified text, added rationale for some standards, updated examples to conform to latest standards
	Change Log	Version 3.1 change log updated--some items were missing

Version	**Section**	**Change**
3.3		Release Date: 6/1/99
	1.0	Added DVD as new medium
	1.3	Changed Version to 3.3
	1.6	Updated/corrected references
	1.7	Added reference to PDS web page
	2.0	Added definition for IAU
		Clarified text
	2.3	Corrected punctuation
	2.7	Fixed punctuation for references
	3.4	Corrected punctuation
	3.7	Corrected spelling and punctuation
	4.0	Added Section headers for Primary & Secondary Objects
	4.1	Corrected paragraph formatting
	5.1.2	Added paragraph about ASCII character set
		Added paragraph about Label Padding
		Fixed math in calculating start byte of 8th record
		Aligned keyword/values
	5.2.2	Corrected grammar
	5.2.3	Removed "’"in the Data Set catalog template.
	5.3.1	Changed Version to 3.3
	5.3.2	Modified last paragraph
	5.3.3	Listed examples of primary and secondary objects
	5.3.3.2	Changed ’bottom’ to ’following’
	5.3.4	Removed AMMOS as an example
	5.3.4.1	Removed SPACECRAFT_NAME as valid keyword
	5.3.4.3	Removed SPACECRAFT_NAME as valid keyword.
	5.3.5	Changed PDS has developed and continues to develop...
		Added example for a pointer (^DESCRIPTION)
	5.3.6	Aligned keyword/values
		Clarified statement
	5.3.7	Changed: needed for conformance
	6.0	Prioritized organizations that PDS works with
	6.1	Provided definition for Data Set Collection and removed MGN example.

	Corrected spelling (considerations) and punctuation
6.2	Added acronyms for data set name and identifier
6.3	Changed paragraph from future tense to past tense
6.4	Section 5 - comets
	Section 6 - added acronyms to list
	Section 6 - corrected spelling (ephemeris)
	Section 7 - corrected spelling (gravity)
	Section 8 - clarified version number rules
7.0	Updated paragraph
7.1	Clarified statements about date/time formats
7.2.1	Added PDS preference for convention
7.3.1	Corrected grammar
	Reformatted paragraph
7.3.2	Corrected grammar
	Updated paragraphs
8.1	Corrected grammar (standards directory)
	Added EXTRAS directory
	Added Browse and Data directory descriptions
8.2	Section 4 - Better examples of directory names
	Section 5 - Reformatted paragraph
	Section 8 - Corrected spelling and grammar
8.3	Changed to valid keywords
8.4	Corrected grammar (data are)
9.0 - 9.3.3	Complete rewrite of Documentation Standard
	Added HTML standards
10.0 - 10.1	Added ISO 9660 Level 2 description
	Added ";1" to Level 1 description
10.2.1	Clarified required file names paragraphs
	Added TARGET_CATALOG pointer to list
10.2.2	VOLDESC.SFD file becomes deprecated
10.2.3	Described detached label
	Corrected grammar (its)
10.2.4	Added extensions and changed SPICE extensions
	Corrected spelling (postscript) and grammar (data that have)
11.1.1	Changed chapter name
12.1	Aligned equal signs
12.1.1.1	Added reference
12.2	Reformatted paragraph
12.3	Spelling
12.3.1	Corrected punctuation (1.234E2)
12.3.1.2	Corrected value (16#+4B#)
	Reformatted paragraph
12.3.1.3	Corrected value (1.234E3)
12.3.2	Updated paragraphs
12.3.2.1	Clarified date format
12.3.2.3	Clarified paragraph

12.3.2.4	Changed year to 4 digits
12.3.2.5	Updated paragraph
12.3.2.5.1	Corrected value (1990-158T15:24:12z)
12.3.3.1	Corrected value ("::=")
12.3.4	Added examples
12.3.5	Corrected punctuation and grammar (units)
12.4	Corrected punctuation
12.4.1	Corrected grammar (the the)
	Aligned equal signs
12.4.2	Aligned equal signs
12.5.2	Reformatted asterisks to not be superscript
	Corrected value (60.15)
12.5.3.1	Corrected grammar (affect)
	Reformatted paragraphs
12.5.4	Corrected value (IO)
	Added valid quoted strings
12.5.4.1	Clarified paragraph
12.5.5	Reformatted asterisk to not be superscript
	Corrected spelling (eccentricity)
	Changed to valid keyword
12.5.6	Corrected value (removed 1st bracket "[")
	Changed to valid keyword
12.6	Reformatted paragraphs
12.7	Reformatted paragraphs
12.7.1	Corrected grammar (sections detail)
12.7.2	Corrected grammar ("is that are")
13.1	Added required keywords to definition
14.1.1	Corrected grammar (occurs)
14.1.2	Corrected punctuation
	Corrected value (^STRUCTURE)
	Changed paragraph numbering
14.2	Reformatted pointer rules
15.0	Reformatted paragraph and table
15.2	Changed paragraph numbering
15.3	Changed paragraph numbering
16.0	Corrected grammar
16.2	Clarified paragraph
	Changed case of #mark#
17.1	Changed case of title (and)
17.1.2	Corrected punctuation (information)
17.2	Corrected case of title (and)
18.0	Corrected SI Units (electricity potential, etc)
	Updated paragraph
19.1	Corrected grammar (volume types)
	Corrected grammar (up to the)
19.3	Corrected grammar (an SFDU)

	Corrected spelling (global)
	Updated Catalog and Index definitions
	Added description of the EXTRAS directory
	Added Preferred Method for supplying PDS catalog objects
19.4.1	Corrected grammar (data have been)
	Changed case of value (ID)
19.5	Corrected spelling (radiometry)
	Corrected value (VOLUME_SET_NAME)
	Corrected value (VOLUME_SET_ID)
19.5.1	Reformatted paragraph
19.7	Corrected case of value (IDs)
20.0 - 20.6	Complete rewrite of Zip Compression
Appendix A	Added URL to Cold Fusion pages
A.1	Updated definition for ALIAS
	Corrected spelling (subobject)
A.2	Added and changed Optional keywords
	Reformatted paragraphs
	Corrected spelling (the time)
A.3	Changed Optional keywords
	Corrected spelling (created)
A.5	Added TARGET to Optional Objects
	Clarified use of CATALOG.CAT
	Formatted paragraph
A.7	Formatted paragraph
	Changed Optional keywords
A.8	Updated paragraph
A.10	Changed case of keyword values to uppercase
A.11	Corrected grammar (on a)
	Corrected grammar (on the medium)
A.12	Removed incorrect statements
	Updated example
A.13	Changed Optional keywords
A.14	Removed a Required keyword
	Added Optional keywords
A.15	Changed value to keyword (GAZETTEER_TABLE)
	Corrected grammar (the breath & upper right)
	Added Optional Keywords section
	Added Optional Objects section
	Added trailing double quote to DESCRIPTION section
A.16	Corrected paragraph to reflect proper file name
	Changed value to be enclosed in double quotes
A.18	Added Required and Optional Keywords and Objects sections
A.19	Added BAND_NAME keyword
	Added Optional keyword
	Changed values to be keyword (CHECKSUM)
	Changed values to be keyword (SCALING_FACTOR)

A.20	Changed paragraphs
	Changed case of keyword values to uppercase
A.21	Reformatted paragraphs
	Removed Optional Keyword
	Added Optional Objects
	Corrected example (see additional example in A.27.1)
A.23	Added example for CORE_ITEM_TYPE
	Corrected FILE_RECORDS to be accurate
	Corrected invalid keyword (SUB_SOLAR_AZIMUTH)
A.24	Corrected grammar (data that vary)
A.26	Corrected grammar (data are)
	Corrected punctuation (The Tookit)
A.27	Corrected grammar (meta-data which are)
	Updated section numbers to reflect location (spares)
	Repaired examples (byte lengths)
A.28	Line length to 72 chars
	Added Required and Optional Objects
	Repaired example
A.29	Updated Optional keyword
	Changed case of keyword values to uppercase
Appendix B	Changed paragraph
	Changed text description length to be 80 characters from 72
	Added text formatting standards
B.1	Corrected punctuation
	Repaired example
B.2	Reformatted paragraph
	Reformatted and repaired example
B.3	Corrected spelling (DESCRIPTION)
	Reformatted paragraph
	Reformatted and repaired example
B.4	Corrected spelling (description & instrument)
	Reformatted paragraph
	Reformatted and repaired example
B.5	Corrected grammar (properties of the)
	Reformatted paragraph
	Reformatted and repaired example
B.6	Repaired example
B.7	Reformatted paragraph
	Reformatted and repaired example
B.8	Repaired example
B.10	Corrected spelling (package)
	Replaced example of SOFTWARE_INVENTORY template
B.11	Corrected grammar (target catalog)
	Corrected grammar (SURFACE_GRAVITY)
	Repaired example
Appendix C	Minor corrections throughout text

	C.5	Corrected spelling (exponent-as-stored)
	C.10	Corrected spelling (imaginary)
	Appendix E	Corrected sentence (source code for)
		Corrected spelling (spacit)
		Corrected grammar (These data are)
		Corrected punctuation
	Appendix F	Corrected CD-WO nomenclature
		Added DE (Data Engineer)
		Corrected spelling (Principal)
	Appendix G	Added SAVED Data as new section

Version	**Section**	**Change**
3.4		Release Date: 06/15/2001

Technical editing of the entire document (Chapters 1-20, Appendices A-G) was performed by Anne Raugh under contract to JPL. This editing focused on correcting awkward language, making examples consistent with the text, clarifying apparent internal inconsistencies, and in general ensuring a more readable document. Substantive changes to the standards themselves were specifically prohibited. Document changes made by Raugh were reviewed by Lyle Huber (ATMOS) and Ron Joyner (CN). Cases in which the intention of the original document could not be determined by the above team were referred to Steve Hughes (CN), who acted as both historian and final arbiter.

On May 04, 2001, Ann Raugh, Richard Simpson, Lyle Huber, Steve Hughes, and Ron Joyner met at New Mexico State University to discuss and arbitrate the final set of changes to be incorporated into this document.

Version	**Section**	**Change**
3.5		Release Date: 10/15/2002
	19.4.1	Changed length of VOLUME_ID from 9 chars to 11 chars
	19.5.1	Changed formation rule for VOLUME_SET_ID
	B.1.6	Modified to include ARCHIVE_STATUS keyword
	B.7.1	Modified to include ARCHIVE_STATUS keyword
		Modified to include DATA_SET_TERSE_DESC keyword
	B.31	Amended Reference section to include more definitive language on what is appropriate to cite, what is not, and how to cite each type of reference.

Version	**Section**	**Change**
3.6		Release Date: 08/01/2003
	3.1	Modified to include FIELD data element
	3.2	Modified to include FIELD data element
	3.8	Modified to include SPREADSHEET object

	4.0	Modified to include SPREADSHEET as primary object
	5.5	Added definition for Locally Defined Data Elements
	5.5.1	Added Justification for Locally Defined Data Elements
	5.5.2	Added Identification for Locally Defined Data Elements
	5.5.3	Added Review and Use of Locally Defined Data Elements
	7.3.1	Modified 1st two paragraphs to clarify GMT/UTC relationship
	7.4(6)	Added Note for Greenwich time
	10.2.3	Modified list to include CSV as reserved file extension
	12.2.3	Modified use of Colon in assignment statements (namespace)
	12.4.2	Modified to include namespace_identifier:element_identifier
	12.6	Modified to include namespace_identifier:element_identifier
	19.3.3.2	Modified to include Optional use of Data Dictionary Files
	A.2.5	Removed 'Z' from time value
	A.13.2.1	Added Note for PARMS as alias to PARAMETERS group
	A.14	Added FIELD object (sub-object of SPREADSHEET)
	A.15 thru A.26	Renumbered sections (the old A.14 became A.15, etc)
	A.15.5	Removed 'Z' from time value
	A.17.5	Removed 'Z' from time value
	A.25	Modified UTC / GMT for LEAPSECONDS
	A.27	Added SPREADSHEET object
	A.23.4	Removed 'Z' from time value
	A.24.5	Removed 'Z' from time value
	A.25.5	Removed 'Z' from time value
	A.26.5	Removed 'Z' from time value
	A.28 thru A.30	Renumbered sections (the old A.27 became A.28, etc)
	A.28.4.1	Removed 'Z' from time value
	A.28.5.1.3	Removed 'Z' from time value
	B.1.6	Modified to include CITATION_DESC data element
	B.7.1	Modified to include CITATION_DESC data element
	B.7.5.3	Modified to include CITATION_DESC formation rule
	B.7.5.4	Renumbered from B.7.5.3

Version	**Section**	**Change**
3.7		Release Date: 03/20/2006

This update of the document focused almost entirely on updates to standards in response to approved Standards Change Requests (SCRs). A few typographical errors were also fixed.

	1.7	Changed hyperlink (http://pds.jpl.nasa.gov) to regular text format.
	3	Corrected chapter title in header on even numbered pages by changing "Definitions" to "Values".
	4	Corrected chapter title in header by changing "Data Products" to "Data Objects and Products".
	4	Added "QUBE" to list of primary data objects.
	5.2.1	Updated Figure 5.2 to include DD_VERSION_ID in response

	to SCR 3-1021; also added LABEL_REVISION_NOTE.
5.2.2	Updated Figure 5.3 to include DD_VERSION_ID in response to SCR 3-1021; also added LABEL_REVISION_NOTE and corrected a few typographical errors ("FILE_RECORD" to "FILE_RECORDS", "Detached" to "detached", spaces inserted before and after "/", alignment of bullets corrected).
5.2.3.1	Changed "identifier" to "identifiers".
5.2.3.2	Changed "identifier" to "identifiers". Modified Figure 5.4 to match format of Figures 5.2 and 5.3; added DD_VERSION_ID in response to SCR 3-1021 and added LABEL_REVISION_NOTE.
5.3.1	Added new paragraph describing DD_VERSION_ID and added keyword to two examples in response to SCR 3-1021.
7.3.2	Changed point 6 to disallow alternate zones in response to SCR 3-1023.
8.2	Modified point 2 to clarify ISO 9660 Level 2 usage in response to SCR 3-1006.
8.3	Modified first paragraph from “Level 1” to “Level 2” and “eight characters” to “31 characters” in response to SCR 3-1006.
10	Deleted “only” from third paragraph in response to SCR 3-1006.
10.1.1	Removed final sentence of first paragraph in response to SCR 3-1006.
10.1.2	Modified first paragraph from "with one exception:" to "with the exception that" and fourth paragraph from "file name specification" to "file and directory name specifications" in response to SCR 3-1006.
10.2.3	Updated IMQ definition to indicate exception for JPEG 2000 images, and added JP2 definition in response to SCR 3-1003.
12.4.5	Removed point 2 which precluded the use of GROUPs within OBJECTs in response to SCR 3-1037. Fixed typographical error in point 16 (“AnEND” to “An END”)
12.7.3	Modified point 14 to provide additional clarification and reference to chapter 7 in response to SCR 3-1023. Fixed typographical error in point 16 (“AnEND” to “An END”)
13	Fixed header by adding "/ Groups" to chapter title.
13.2	Removed page break before 13.2.
13.2.1	Removed point 2 which precluded the use of GROUPs within OBJECTs in response to SCR 3-1037.
20	Chapter 20 (Zip Compression) is now included as section 6 of Appendix I (Data Compression Formats), in response to SCR 3-1003.
A	Corrected spelling of "SHREADSHEET" in chapter contents.
A.25	Added new section describing SPECTRAL_QUBE object in response to SCR 3-1037.
A.26-A31	Appendices A.25 through A.30 renumbered in response to SCR 3-1037.
A.28	Corrected spelling of section title from "SHREADSHEET" to "SPREADSHEET".
B.1.3	Modified to include DATA_SET_MISSION catalog object in response to SCR 3-1028.
B.1.6	Modified to include DATA_SET_MISSION catalog object in

		response to SCR 3-1028. Updated example of DATA_SET catalog object to include ABSTRACT_DESC in response to SCR 3-1026.
	B.7.1	Updated DATA_SET_INFORMATION catalog object to include required keyword ABSTRACT_DESC in response to SCR 3-1026.
	B.10	Added DATA_SET_MISSION catalog object in response to SCR 3-1028.
	B.15	Added a sentence regarding multiple instrument hosts to the first paragraph in response to SCR 3-1024.
	B.15.5	In the second sentence, change “A” to “An”.
	B.32.5.6.1	Modified description of author list in REFERENCE_DESC in response to SCR 3-1005.
	D	Changed chapter contents from hyperlinks to plain text.
	E	Fixed header on even-numbered pages by moving page number to left side and chapter title to right side.
	H.1	Added new section describing BAND_BIN group in response to SCR 3-1037.
	H.2	Added new section describing BAND_SUFFIX group in response to SCR 3-1037.
	H.3	Added new section describing LINE_SUFFIX group in response to SCR 3-1037.
	H.4	Renumbered from H.1 in response to SCR 3-1037.
	H.5	Added new section describing SAMPLE_SUFFIX group in response to SCR 3-1037.
	I	Added Appendix I (Data Compression Formats) in response to SCR 3-1003. New appendix includes former chapter 20 (Zip Compression).

Version	**Section**	**Change**
3.8		Release Date: 02/16/2009

With the exception of the changes to chapter 2, this update of the document focused almost entirely on updates to standards in response to approved Standards Change Requests (SCRs).

	1.3	Changed 3.4 to 3.8.
	2	Completely re-written to bring PDS cartographic standards up-to-date with underlying reference frames and coordinate systems approved by various external authorities. (Work performed and approved directly by MC.)
	5.3	Added text prohibiting multiple instances of keywords within a given label context in response to SCR3-1126.
	6.3.2	Added Frames Kernel to the list in subsection 6 in response to SCR 3-1051.
	7	Numerous changes made to clarify preferred PDS date/time format and to remove trailing “Z” in response to SCR 3-1103.
	7.4	New section on leap seconds added in response to SCR 3-1103.

8.2	Added sentence to end of point 2 in response to SCR 3-1144.
8.3	Added sentence to middle of first paragraph is response to SCR 3-1144. Added trailing slashes to the examples in response to SCR 3-1107.
10	Added distinction between naming standards for physical and electronic media deliveries and added section entitled "Electronic Transfer and Storage of Archives" in response to SCR 3-1144.
10.1.3	New section added in response to SCR 3-1144.
10.2.3	Added .FIT and .FTS to Table 10.1 and added second paragraph in response to SCR 3-1129. Added .TF to Table 10.1 in response to SCR 3-1051. Modified text to point to Table 10.1 on following page, rather than immediately following the second paragraph. Corrected the reference to chapter 9.
11	First paragraph modified and new section, "Electronic Transfer and Stoage of Archive" added in response to SCR 3-1144.
12.3.2.5.1	Removed the first sentence of the second paragraph in response to SCR 3-1133.
12.4.2	Added text to the first paragraph prohibiting multiple instances of keywords within a given label context in response to SCR3-1126. Also changed "asignment" to "assignment".
12.4.5	Added text to the second restriction prohibiting multiple instances of keywords within a given label context in response to SCR3-1126.
12.5.5	Corrected BNF notation for 2D sequences, fixed example of 1D sequence and added 2D example in response to SCR 3-1087.
19.1	Removed extraneous "and" from point 3b.
Fig. 19.2	Added missing contents of INDEX directory.
19.3.2.1	Under "PDS Methodology for Supplying…", changed refered to Fig. 19.5 to 19.4, and changed file names to match those in figure.
19.3.2.3	Changed reference to appendix A.20 to A.21.
19.3.3.1	Under "Calibration Files", changed references to Figures 19.3 and 19.5 to 19.3a and 19.4, respectively.
19.3.3.4	Changed reference to Appendix A.15 to A.16.
19.3.3.6	Changed references to Figures 19.1-5 to 19.1-4.
19.3.3.7	Under "Hardware Platform and Operating System…", changed references to Figure 19.6 to 19.5.
Fig. 19.6	Renamed Figures 19.6 to 19.5.
A.8.5	In first label example, dropped time portion of value for product_release_date in response to SCR 3-1117.
A.20	Additional text added to final paragraph to clarify usage of LINE_ and SAMPLE_DISPLAY_DIRECTION keywords in response to SCR 3-1130.
A.20.2	Added LINE_ and SAMPLE_DISPLAY_DIRECTION to list of optional keywords in IMAGE object in response to SCR 3-1102.
A.20.4	Added optional sub-object WINDOW to IMAGE object in response to SCR 3-0004.
A.25.4.8	Added text to table column "Values" for keywords LINE_ and SAMPLE_DISPLAY_DIRECTION to clarify their usage in response to SCR 3-1130.

A.27	Added FRAMES kernel to table of kernel types in response to SCR 3-1051.
A.32	Added new section on WINDOW object in response to SCR 3-0004.
B	Changed “72” in fourth paragraph to “80” in response to SCR 3-1110.
B.1.6	Removed “ARCHIVED_STATUS = ARCHIVED” from the example in response to SCR 3-1108.
B.7.1	Removed ARCHIVE_STATUS from list of required keywords in response to SCR 3-1108.
B.14	Improved the first three paragraphs of the section in accordance with SCR 3-1138.
B.14.3	Added optional keywords in response to SCR 3-1138.

(This page intentionally left blank.)

Chapter 1. Introduction

In order for planetary science data to be useful to those not directly involved in its creation, supporting information must be made available with the data to allow effective use and interpretation. The exchange of data is increasingly important in planetary science; thus there is a need for establishment and enforcement of standards regarding the quality and completeness of data. Electronic communication has become more sophisticated, and the use of new media (such as CD-ROMs and DVD) for data storage and transfer requires additional formatting standards to ensure long-term readability and usability. To these ends, the Planetary Data System (PDS) has developed a data set nomenclature consistent across discipline boundaries, as well as standards for labeling data files.

1.1 PDS Data Policy

Only data that comply with PDS standards will be published in volumes labeled "Conforms to PDS Standards". When the PDS assists in the preparation of data published in a non-compliant format, PDS participation should be acknowledged with the statement such as "funded by PDS". The PDS Management Council makes decisions on compliance waivers. Non-compliant data sets will be incorporated into the PDS archives only under unusual circumstances.

1.2 Purpose

This document is intended as a reference manual for use in conjunction with the *PDS Data Preparation Workbook* and the *Planetary Science Data Dictionary*. The *PDS Data Preparation Workbook* describes the end-to-end process for submitting data to the PDS and gives instructions for preparing data sets. In addition, a glossary of terms used throughout the documentation is included as an appendix to the Workbook. The Planetary Science Data Dictionary (PSDD) contains definitions of the standard data element names and objects. This Standards Reference defines all PDS standards for data preparation.

1.3 Scope

The information included here constitutes Version 3.8 of the Planetary Data System data preparation standards for producing archive quality data sets.

1.4 Audience

This document is intended primarily to serve the community of scientists and engineers responsible for preparing planetary science data sets for submission to the PDS. These include restored data from the era prior to PDS, mission data from active and future planetary missions, and data from earth-based sites. The audience includes personnel at PDS discipline and data nodes, mission principal investigators, and ground data system engineers.

1.5 Document Organization

The first chapter of this document, “Chapter 1 – Introduction”, provides introductory material and citations of other reference documents. The remaining chapters provide an encyclopedia of data preparation standards, organized alphabetically by standard title.

1.6 Other Reference Documents

The following references are cited in this document:

- Batson, R. M., (1987) “Digital Cartography of the Planets: its Status and Future”, *Photogrammetric Engineering & Remote Sensing 53,* 1211-1218.
- Davies, M.E., *et al.* (1991) “Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 1991”, *Celestial Mechanics*, 53,377-397.
- Greeley, R. and Batson, R.M. (1990) *Planetary Mapping*, Cambridge University Press, Cambridge, 296p.
- *Guide on Data Entity Naming Conventions,* NBS Special Publication 500-149.
- *Planetary Science Data Dictionary,* JPL D-7116 Rev D, July 15, 1996, (Available from the PDS).
- *Planetary Data System Data Preparation Workbook Version 3.1,* JPL D-7669 Part 1, February 17, 1995, (Available from the PDS)
- *Issues and Recommendations Associated with Distributed Computation and Data Management Systems for the Space Sciences,* National Academy Press, Washington, DC, 111p.

International Standards Organization (ISO) References:

- ISO 9660:1988 “Information Processing - Volume and File Structure of CD-ROM for Information Exchange”, April 15, 1988.
- ISO 646:1991 ASCII character set.
- ISO 8601:1988 “Data Element and Interchange Formats – Representations of Dates and Times”

`SFDU and PVL References:`

- *Standard Formatted Data Units - Structure and Construction Rules,* CCSDS 620.0-R-1.1c, May 1992.

- *Standard Formatted Data Units - A Tutorial;* CCSDS 620.0-G-1, May 1992.
- *Parameter Value Language Specification (ccsd0006);* CCSD 641.0-R-0.2, June 1991.
- *Parameter Value Language -- A Tutorial;* CCSDS 641.0-G-1.0, May 1992.

1.7 Online Document Availability

The *Planetary Science Data Dictionary*, *Planetary Data System Data Preparation Workbook*, and this document, the *Planetary Data System Standards Reference,* are available online. Information on accessing these references may be found on the PDS website at the following URL:

http://pds.jpl.nasa.gov

To obtain a copy of these documents or for questions concerning these documents, contact the PDS Operator (at PDS_OPERATOR@jpl.nasa.gov, 626-744-5579) or a PDS data engineer.

The examples provided throughout the chapters and appendices are based on both existing and planned PDS archive products, modified to reflect the current version of the PDS Standards. Data object definitions are refined and augmented from time to time, as user community needs arise, so object definitions from products designed under older versions of the Standards may differ significantly. To check the current state of any object definition, consult a PDS data engineer or this URL:

http://pdsproto.jpl.nasa.gov/ddcolstdval/newdd/top.cfm

Additional examples may be obtained by contacting a Data Engineer.

(This page intentionally left blank.)

Chapter 2. Cartographic Standards

2.1 Introduction

To facilitate use, exchange and integration of its products, the PDS follows accepted planetary cartographic standards for data products where they exist. Because such standards evolve as new data and knowledge are acquired, there are advisory groups charged with developing and periodically updating standards for coordinate systems. All data providers for PDS products should follow accepted standards and be aware of current NASA and international recommendations on cartographic coordinate systems and conventions relevant to their bodies of interest. ***An absolute requirement for all PDS products is that relevant coordinate systems and frames be clearly specified in product labels and supporting documents.*** This chapter specifies, as of late 2008, the authoritative sources for international cartographic standards, provides a summary of major cartographic elements to which those standards apply, and identifies the primary standards that PDS has adopted.

2.1.1 International and NASA Advisory Groups for Cartographic Standards

The primary international body for coordinate systems in the Solar System is the International Astronomical Union (IAU). The IAU has recognized the International Celestial Reference System (ICRS) as the defining inertial reference system and its associated International Celestial Reference Frame (ICRF) (Ma et al., 1998) as the defining frame for that system. The ICRS and ICRF are maintained for the IAU by the International Earth Rotation and Reference Systems Service (IERS, http://www.iers.org/).

For cartographic coordinates and conventions for planets and satellites, the IAU and the International Association of Geodesy (IAG) have established jointly the Working Group on Cartographic Coordinates and Rotational Elements (WGCCRE), which publishes triennial reports, currently in the journal *Celestial Mechanics and Dynamical Astronomy* (Davies, et al., 1980, 1983, 1986, 1989, 1992, 1996; Seidelmann, et al., 2002, 2005, 2007). This working group includes PDS-affiliated scientists, thus assuring full interaction in defining the standards. Publications and reports issued by the WGCCRE can be found at http://astrogeology.usgs.gov/Projects/WGCCRE/. PDS data providers should refer to these reports for current information and recommendations on rotational elements for Solar System bodies and how these are related to their cartographic coordinates.

The NASA Lunar Geodesy and Cartography Working Group and the Mars Geodesy and Cartography Working Group are sponsored by the NASA Lunar Precursor Robotics Program (LPRP) and Mars Program offices, respectively, and are responsible within NASA for providing additional coordination of cartographic standards and related (e.g., data processing) issues (Archinal et al., 2008a, 2008b; Duxbury et al., 2002). These Working Groups have made additional recommendations regarding coordinate systems (generally with additional detail) beyond those of the WGCCRE.

2.2 Inertial Reference Frame and Time System

The orientation of a body in the Solar System can be calculated using a series of rotation angles to define the directions of the body's principal axes with respect to an inertial reference frame (i.e., a system that is not rotating or accelerating relative to a specific reference point) which provides a standard frame from which position, velocity, and acceleration can be measured. Such a reference frame is a set of identifiable fiducial points and their positions on the sky, providing a practical realization of a reference system that defines the origin, fundamental planes (or axes), and transformations between observed elements and reference points in the celestial coordinate system. Reference coordinate systems are defined by a system of concepts (e.g., using planetocentric latitude and longitude) while a reference coordinate frame is a specific realization of a coordinate system that is anchored to real data (such as a photogrammetric control network, altimetry crossover solutions, or lunar ephemerides) (Kovalevsky and Mueller, 1981).

For a planetary body in space, position is defined relative to a Z axis (typically the spin vector of the body, or the planetographic north pole), the X axis (defined as the point where the equator of the body crosses the equatorial plane of an inertial frame at a specific epoch), and the Y axis of a right-handed system. The standard units for coordinates are based on the International System of Units (SI), including decimal degrees. The orientation of Solar System bodies can be calculated from angular position (right ascension α and declination δ) with respect to the equatorial system of a particular epoch. For example, the orientation of the north pole of a body at a given epoch is specified by its right ascension α and declination δ, while the location of the prime meridian is specified by the angle W (Davies et al., 1980).

The standard epoch is called J2000.0 and is defined to be 2000 January 1.5 TDB, where TDB is Barycentric Dynamical Time (e.g., Seidelmann et al., 2007). This corresponds to 2000 January 1, 1200 hours TT (Terrestrial Time) or the Julian Date 2451545.0 (NAO, USNO and HMNAO, 1983). This also corresponds to 2000 January 1, 11:58:55.816 UTC (Coordinated Universal Time; Seidelmann et al., 1992). Although the natural system for many applications would be TDB, UTC is considered the fundamental system for all PDS data products. The standard way of expressing UTC is in year, month, day, hour, minute, and decimal seconds. Julian Dates (JD) are supported as a supplementary system for reporting UTC time. However the JD time scale must be specified (e.g., UTC or TDB). See the Planetary Science Data Dictionary (PDS, 2008), chapter 2, for further information on time representation.

The currently accepted orientation of the inertial system (i.e., J2000.0 right ascension and declination) is defined by the International Celestial Reference System (ICRS), which is a particular implementation of the Barycentric Celestial Reference System (BCRS) (IAU, 2000). The ICRS is the fundamental celestial reference system of the IAU, and it has an origin at the barycenter of the Solar System and 'space fixed' (kinematically non-rotating) axis directions. As noted by the IAU, the ICRS is meant to represent the most appropriate coordinate system for expressing reference data on the positions and motions of celestial objects. Specifications for the ICRS include a metric tensor, a prescribed method for establishing and maintaining axis directions, a list of benchmark objects with precise coordinates, and standard algorithms to transform these coordinates into observable quantities for any location and time. The ICRS is derived from the International Celestial Reference Frame (ICRF) comprised of coordinates for a set of fiducial points on the sky. The ICRF is within 0.05 arcseconds (Chapront et al., 2002; Herring et al., 2002) of the Solar System inertial frame based on Earth's Mean Equator (EME) at the Equinox of Julian Ephemeris Date (JD) 2451545.0 (i.e., J2000.0). This is consistent with current dynamical practice and spacecraft and planetary ephemerides (e.g., those provided by the NASA Jet Propulsion Laboratory).

Many older data sets, collected before the J2000.0 system and ICRF were defined, are referenced to EME and Equinox of Besselian 1950.0 (B1950.0; JD 2433282.423). While this reference frame should not be used for current data, PDS supports this reference frame for older data. Transformation between the "B1950.0" and "J2000.0" (and the nearly equivalent ICRF) systems has been well defined by the IAU (NAO, USNO and HMSNAO, 1983; also see http://nedwww.ipac.caltech.edu/forms/calculator.html).

Positions may be expressed in other coordinate systems and associated frames, which can be derived from the fundamental system and frame, when this enhances the use of the data for various applications. These include ecliptic-based coordinates and heliographic coordinates. These coordinates, while possibly "natural" for many applications, are derivable from the fundamental system and are therefore treated as supplementary data by PDS. In some cases, it is convenient to work in one preferred coordinate system and then to convert to another, more standard system for products. This practice of providing the natural working coordinates in addition to the coordinates in a fundamental system promotes ease of use of PDS products and should be adopted by all data providers who use coordinate systems other than the fundamental system. As noted above, all supplementary coordinate systems must be fully documented in PDS products and must be negotiated with the PDS prior to delivery.

2.3 Spin Axes and Prime Meridians

The spin axis orientations of many Solar System bodies are defined by the WGCCRE in the ICRF inertial reference frame. For historical reasons, the orientation of the spin axis of planets and satellites is defined by the "north" pole, which is the pole that is on the northern side of the Invariant Plane of the Solar System (close to but not the same as the ecliptic). With this definition of the north pole, it is also necessary to specify whether the rotation is direct or 'prograde' (in the same direction as the Sun's rotation or counterclockwise when viewed from above the north pole) or retrograde (opposite to the direction of the Sun's rotation).

For small bodies such as comets and asteroids, for which precession due to torques can cause large changes in the angular momentum vector, the orientation is defined by the 'positive' pole, which is the pole determined by the right hand rule for rotation. Since some small bodies can be in excited state rotation, there are numerous complications in application that are addressed in more detail in the WGCCRE reports. Depending on the mode of excited state rotation, the axis may coincide with the maximum moment of inertia. Some cases, particularly the case of chaotic rotation, are considered on a case by case basis by the WGCCRE.

If a body has a solid surface, prime meridians for a given longitude system may be defined by specifying the coordinates of a surface feature on the body (usually a small feature such as a crater in the equatorial region) or by the mean direction relative to the parent body for synchronously rotating bodies (e.g., the Moon, the Galilean moons, and most of the Saturnian moons). Where insufficient observations exist to determine the principal moment of inertia, coordinates of a surface feature will be specified and used to define the prime meridian. In the case of planets without solid surfaces, the definition of the prime meridian is somewhat arbitrary. In any case, the actual definitions are decided by the WGCCRE, not by the PDS. We note that influxes of new data often lead to an iterative process to define (or improve) the orientation of the spin axis or other parameters used to define a coordinate system and in these cases the data providers (e.g., spacecraft mission personnel) and the WGCCRE must maintain close contact regarding the definition.

2.4 Body-Fixed Planetary Coordinate Systems

Two types of coordinate systems are fixed to the body – planetocentric and planetographic. Details of the coordinate systems for planets and satellites differ from those for small bodies and rings. This section discusses only the aspects that are common to all applications. The ***Planetocentric*** system has an origin at the center of mass of the body. Planetocentric coordinates are defined by a vector from the center of mass of the body (often approximated as the center of figure) to the point of interest, typically but not necessarily a point on the surface (e.g., an impact crater with known position). The planetocentric latitude is the angle between the equatorial plane and the vector, while the planetocentric longitude is the angle between the prime meridian and the projection of the vector onto the equatorial plane.

The ***Planetographic*** system also has an origin at the center of mass of the body. Planetographic coordinates, however, are defined by vectors perpendicular to a reference surface, often a biaxial ellipsoid that is centered on the body and chosen to describe the gross shape of the body. Reference surfaces vary from body to body and are defined by the WGCCRE in consultation with the observers who provide the information to define such surfaces. The most common reference surface is an oblate spheroid aligned with the spin axis of the body. However, for certain applications the reference surface may be a triaxial ellipsoid, a gravitational equipotential, or a higher order surface model.

For a biaxial ellipsoid the planetographic latitude is the angle between the equatorial plane and a vector through the point of interest, where the vector is normal to the reference surface. Planetographic longitude is the angle between the prime meridian and the projection of the same vector onto the equatorial plane. In general, the planetographic vector does not pass through the origin. The vector need not pass through the spin axis but in most realistic cases it does. If the reference surface is a sphere, the planetographic and planetocentric vectors are identical.

The WGCCRE allows for the use of either planetographic or planetocentric coordinates for a given body, so data providers may adopt either system. Historically planetographic coordinates have been preferred for cartographic products, while planetocentric coordinates were used for dynamical (i.e. orbit, gravity field, altimetric) observations and calculations. For the planet Mercury, the MESSENGER mission has chosen to use planetocentric coordinates as the primary coordinate system for all products (Seidelmann et al., 2007). For the planet Mars, the MGCWG and all current NASA missions have chosen to use planetocentric coordinates as the primary coordinate system for products (Duxbury et al., 2002). Producers of printed or electronically printed maps (e.g., in PDF format) may wish to show both types of coordinates.

2.4.1 Planets and Satellites

For planets and satellites, the conventions are complicated for historical reasons. In the planetocentric coordinate system, northern latitudes are those in the hemisphere of the body containing the spin pole that points to the northern side of the invariant plane of the Solar System. The body's rotation direction, either prograde or retrograde, must also be specified. Planetocentric longitude increases eastward (i.e., in the direction defined by the right-hand rule and the "north" pole) from the prime meridian, from 0° to 360°. Thus an external observer sees the longitude decreasing with time if the rotation is prograde but increasing with time if the rotation is retrograde.

North and south planetographic latitude are defined in the same way as for planetocentric latitude, although the numerical values for a given point on the surface, (other than on the equator or at the poles) are different if the reference surface is not a sphere. The definition of planetographic longitude is dependent upon the rotation direction of the body, with the basic definition being that an external observer should see the longitude increasing with time, or that the longitude increases in the direction opposite to the rotation, although there are exceptions due to historical practice for Earth, the Moon, and Sun. That is to say, the longitude increases to the west if the rotation is prograde (or eastward) and vice versa. Whether the rotation direction is prograde or retrograde can be determined from the current WGCCRE report. See Tables 1 and 2 (or their equivalent in any future report), where the sign of the velocity term for W indicates either prograde (positive) or retrograde (negative) rotation. For all bodies a longitude range of 0° to 360° can be used.

For Earth, the Moon, and the Sun, a longitude range of -180° to +180° has been used in the past [including in existing PDS data sets, as defined by the *Planetary Science Data Dictionary* (PDS, 2002)] and is allowed by the WGCCRE. However, for the Moon, the NASA LGCWG and LRO Mission recommend that in the future, only the 0° to 360° range be used (LGCWG, 2008; LRO Project, 2008). For printed or electronically printed maps (e.g., in PDF format), it may be useful to label the longitude grid both with primary 0° to 360° coordinates and -180° to +180° coordinates.

For the Moon, two slightly different reference systems are commonly used to orient the lunar body-fixed coordinate system. One is the Mean Earth/Polar Axis (ME) system, the preferred system to be used for PDS data products. The other is the axis of figure system, also called the Principal Axis (PA) system, sometimes used internally among instrument teams for specific applications. For computing precise lunar coordinates, the WGCCRE recommends the use of the JPL DE403 ephemeris (which provides lunar orientation in the PA system), rotated into the ME system. The WGCCRE noted in its most recent report that improved versions of the JPL ephemerides were imminent and might be used instead. In fact the JPL DE421 ephemeris is now available and, after rotation into the ME system, is recommended for use (LGCWG, 2008; LRO Project, 2008). The maximum difference between these two frames in the ME system for the period 2000-2019 is only about 6 meters (Archinal, 2008).

2.4.2 Small Bodies

For small bodies (asteroids and comets), both planetographic and planetocentric coordinates follow the same right hand rule that is used to define the positive pole, which can be either above or below the invariant plane of the Solar System. For the simple case of a body with positive pole pointing to the northern hemisphere of the Solar System, this corresponds to longitude, both planetocentric and planetographic, increasing eastward, 0° to 360°, which in turn corresponds to the case in which the longitude seen by an outside observer decreases with time.

For some small bodies, coordinates based on latitude and longitude alone can be multi-valued in radius — i.e., the vector from the center of the body can intersect the surface in more than one place. There may also be complications (due to the irregular shape) which force special procedures when producing a useful, planar map. Such details are discussed in reports of the WGCCRE.

2.4.3 Rings

There is no international standard for ring coordinate systems. Standards in use for such PDS products were defined by experts in the Rings Node, in consultation with a broad cross-section of interested scientists. Conventions for coordinate systems for rings are similar to those for small bodies, in as much as they are all based on a right-hand rule, with longitude increasing in the direction of orbital motion. Thus longitude increases eastward for the prograde-moving rings (Jupiter, Saturn, and Neptune), but it increases westward for retrograde-moving rings of Uranus. Rings also use a positive pole direction following the right hand rule, analogous to the case for small-body rotation, thus coinciding with the North Pole of Jupiter, Saturn, and Neptune, but the South Pole of Uranus.

Coordinates for rings differ from those for planets and small bodies in not being body-fixed because there are no fixed features to define longitude. They are defined in an inertial system that is co-moving with the center of mass of the parent body. Specifically, longitudes are measured from the ascending node of the plane of the rings in the ICRF, i.e. the point at which the plane of the rings intersects the ICRF equator. In the case of inclined rings, longitudes are measured as a “broken angle” from the ascending node of the planet’s equatorial plane in the ICRF, along the equatorial plane to the ring plane’s ascending node, and thereafter along the ring plane.

2.4.4 Planetary Plasma Interactions

There are no international standards for values or names of coordinate systems of planetary plasma observations. Recommendations for coordinate systems in the near-Earth environment by Russell (1971) have been generalized for use with plasma observations at other bodies. More recently, other systems have been defined (e.g., Franz and Harper, 2002) and are currently in use. The coordinate systems used for plasma observations and data analysis typically are right-handed. The primary exception to this rule is the left-handed Jovian System III.

Standards for planetary plasma data products for PDS were defined by experts in the Planetary Plasma Interactions Node, following recommendations from Russell (1971) and Franz and Harper (2002) and in consultation with other specialists. Providers and users of PDS data featuring plasma observations are encouraged to use names as defined by these authors where appropriate, and to follow similar name construction when new systems must be defined.

2.5 Surface Models

A standard reference surface model commonly used for hard surfaces is the digital terrain model (DTM). The DTM defines body radius or geometric height above the body reference surface as a function of cartographic latitude and longitude. Spheroids, ellipsoids and harmonic expansions giving analytic expressions for radius as a function of cartographic coordinates are all allowed in PDS. A DTM may also define potential height, i.e., “elevation”, above an equipotential surface, *provided* the method is specified, including the specification of appropriate constants and gravity field that is used to convert to/from radii and potential height.

The only internationally recognized DTM is the MOLA model for Mars (Seidelmann, et al., 2007, page 168 in WGCCRE #10). DTMs are also available for other bodies, including the Moon and several small bodies; but their use is not officially recommended and therefore up to the individual user.

The digital image model (DIM) defines body brightness in a specified spectral band or bands as a function of cartographic latitude and longitude. A DIM may be associated with the surface radius, geometric height, or potential height values in a corresponding DTM or it may be registered independently to a spheroid, ellipsoid, or spherical harmonic expansion.

2.6 PDS Keywords for Cartographic Coordinates

To support the descriptions of these various reference coordinate systems and frames, the PDS has defined the following set of ‘geometry’ data elements [see the *Planetary Science Data Dictionary* (PDS, 2008) for complete definitions and additional data elements].

A_AXIS_RADIUS
B_AXIS_RADIUS
C_AXIS_RADIUS
COORDINATE_SYSTEM_CENTER_NAME
COORDINATE_SYSTEM_DESC
COORDINATE_SYSTEM_ID
COORDINATE_SYSTEM_NAME
COORDINATE_SYSTEM_REF_EPOCH
COORDINATE_SYSTEM_TYPE
EASTERNMOST_LONGITUDE
LATITUDE
LONGITUDE
MAXIMUM_LATITUDE
MAXIMUM_LONGITUDE
MINIMUM_LATITUDE
MINIMUM_LONGITUDE
POSITIVE_LONGITUDE_DIRECTION

WESTERNMOST_LONGITUDE

To support the description of locations in a planetary ring system, the PDS has defined the following data elements:

CENTER_RING_RADIUS
RING_RADIUS
MINIMUM_RING_RADIUS
MAXIMUM_RING_RADIUS

RING_LONGITUDE
MINIMUM_RING_LONGITUDE
MAXIMUM_RING_LONGITUDE

B1950_RING_LONGITUDE
MINIMUM_B1950_RING_LONGITUDE
MAXIMUM_B1950_RING_LONGITUDE

RING_EVENT_TIME
RING_EVENT_START_TIME
RING_EVENT_STOP_TIME

RADIAL_RESOLUTION
MINIMUM_RADIAL_RESOLUTION
MAXIMUM_RADIAL_RESOLUTION

The radius and longitude elements define an inertial location in the rings, and the ring event time elements define the time at the ring plane to which an observation refers. If desired, the radial resolution elements can be used to specify the radial dimensions of ring features that can be resolved in the data. See the Planetary Science Data Dictionary (PSDD; PDS, 2008) for complete definitions of these elements.

Some rings are not circular and/or equatorial. In these cases, the PSDD provides additional elements that can be used to describe a ring's shape. The elements are:

RING_SEMIMAJOR_AXIS
RING_ECCENTRICITY
RING_PERICENTER_LONGITUDE
PERICENTER_PRECESSION_RATE
RING_INCLINATION
RING_ASCENDING_NODE_LONGITUDE
NODAL_REGRESSION_RATE
REFERENCE_TIME

Here the value of REFERENCE_TIME indicates the instant at which the LONGITUDE elements are defined. The actual pericenter and ascending node at the time of an observation are determined based on the precession and regression rates as follows:

pericenter_longitude = RING_PERICENTER_LONGITUDE +
PERICENTER_PRECESSION_RATE *
(observation_time - REFERENCE_TIME) mod 360
ascending_node_longitude =
RING_ASCENDING_NODE_LONGITUDE +
NODAL_REGRESSION_RATE *
(observation_time - REFERENCE_TIME) mod 360

The oscillating modes of a ring can also be specified if necessary:

RING_RADIAL_MODE
RING_RADIAL_MODE_AMPLITUDE
RING_RADIAL_MODE_FREQUENCY
RING_RADIAL_MODE_PHASE

Additional elements should be used to specify the assumed orientation of the planet's pole:

POLE_RIGHT_ASCENSION
POLE_DECLINATION
COORDINATE_SYSTEM_ID

The COORDINATE_SYSTEM_ID can be either "J2000.0" or "B1950.0", with "J2000.0" serving as the default. See the PSDD for further details.

2.7 Map Resolution

A uniform set of resolutions is helpful for analyses of multiple datasets and development of map products derived from PDS data, and the selected scale must account for differences in available image resolution and quality. Such map scales are measured against a reference surface that is typically a geometrically defined shape that represents a given planetary body. For global maps, the recommended spatial resolution for a map is 2^n pixels per degree of latitude, where a pixel is treated as a finite area and n is an integer. A spatial resolution of 2^n pixels per degree allows simple coregistration of multiple datasets by doubling or halving the pixel sizes (typically by averaging or interpolation) and without resampling or otherwise changing the pixels. These recommendations continue a convention established in the 1960s and 1970s by the lunar and Mars research communities (e.g., Batson, 1987; Greeley and Batson, 1990), as advocated by the NASA Planetary Cartography Working Group (PCWG) and its successor the Planetary Cartography and Geologic Mapping Working Group (PCGMWG) (PCWG, 1993, pp. 22-24), and affirmed by the LGCWG (2008).

For polar regions of global maps, the recommendation is also to use the binary map scale or 2^n pixels per degree of latitude near the pole. This practice maintains consistency with the global data product.

For working at landing site scales with data that has pixels of tens of centimeters to a few meters in size, spatial resolutions of maps are more convenient if provided at scales of 1 meter per pixel resolution or multiples thereof (LGCWG, 2008). At such human scales this convention is simpler and will preserve inherent details of resolution for applications such as landing site operations, traversing, and surface engineering studies.

For both global and local maps showing elevation or relief, the recommended vertical resolution is 1 x 10^m meters, where m is an integer chosen to preserve all the resolution inherent in the data.

2.8 References

(Note: All WGCCRE reports are listed below for completeness. WGCCRE Report 7 was not issued).

Archinal, B. A. (2008). "Summary of Lunar Geodesy and Cartography Working Group Teleconference of Tuesday, 2008 April 16", May 18.

Archinal, B. A. and the Lunar Geodesy and Cartography Working Group (2008a). "Lunar Mapping Standards and the NASA LPRP Lunar Geodesy and Cartography Working Group," Scientific Event B01, "The Moon: Science, New Results, Ongoing Missions, Future Robotic and Human Exploration,", 37th COSPAR Scientific Assembly, July 13-20, Montreal, Canada, http://www.cospar-assembly.org/home.php. Abstract B01-0050-08 and available at http://www.cospar-assembly.org/user/download.php?id=3015&type=abstract&section=congressbrowser

Archinal, B. A. and the Lunar Geodesy and Cartography Working Group (2008b). "Lunar Science Support Activities by the NASA LPRP Lunar Geodesy and Cartography Working Group: Recommendations for Lunar Cartographic Standards," NLSI Lunar Science Conference, July 20-23, Moffett Field, CA. Abstract no. 2080, available at http://www.lpi.usra.edu/meetings/nlsc2008/pdf/2080.pdf.

Batson, R. M. (1987) "Digital Cartography of the Planets: New Methods, its Status and Future," Photogrammetric Engineering & Remote Sensing, 53, 1211-1218.

Chapront, J., Chapront-Touzé, M., & Francou, G. (2002). A new determination of lunar orbital parameters, precession constant and tidal acceleration from LLR measurements, Astronomy and Astrophysics, 387, 700-709.

Davies, M. E. (Chair), V. K. Abalakin, C. A. Cross, R. L. Duncombe, H. Masursky, B. Morando, T. C. Owen, P. K. Seidelmann, A. T. Sinclair, G. A. Wilkins, and Y. S. Tjuflin (1980). "Report of the IAU Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites," Celestial Mechanics, 22, pp. 205-230. [WGCCRE #1]

Davies, M. E. (Chair), V. K. Abalakin, J. H. Lieske, P. K. Seidelmann, A. T. Sinclair, A. M. Sinzi, B. A. Smith, and Y. S. Tjuflin (consultant) (1983). "Report of the IAU Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 1982," Celestial

Mechanics, 29, pp. 309-321. [WGCCRE #2]

Davies, M. E. (Chair), V. K. Abalakin, M. Bursa, T. Lederle, J. H. Lieske, R. H. Rapp, P. K. Seidelmann, A. T. Sinclair, V. G. Teifel, and Y. S. Tjuflin (consultant) (1986). "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 1985," Celestial Mechanics, 39, pp. 103-113. [WGCCRE #3]

Davies, M. E. (Chair), V. K. Abalakin, M. Bursa, G. E. Hunt, J. H. Lieske, B. Morando, R. H. Rapp, P. K. Seidelmann, A. T. Sinclair, and Y. S. Tjuflin (consultant) (1989). "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 1988," Celestial Mechanics and Dynamical Astronomy, 46, pp. 187-204. [WGCCRE #4]

Davies, M. E. (Chair), V. K. Abalakin, A. Brahic, M. Bursa, B. H. Chovitz, P. K. Seidelmann, A. T. Sinclair, and Y. S. Tjuflin (consultant) (1992). "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 1991," Celestial Mechanics and Dynamical Astronomy, 53, pp. 377-397. [WGCCRE #5]

Davies, M. E. (Chair), V. K. Abalakin, M. Bursa, J. H. Lieske, B. Morando, D. Morrison, P. K. Seidelmann (Vice-Chair), A. T. Sinclair, B. Yallop, and Y. S. Tjuflin (consultant) (1996). "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 1994," Celestial Mechanics and Dynamical Astronomy, 63, pp. 127-148. [WGCCRE #6]

Duxbury, T. C., R. L. Kirk, B. A. Archinal, and G. A. Neumann (2002). "Mars Geodesy/Cartography Working Group Recommendations on Mars Cartographic Constants and Coordinate Systems," ISPRS, v. 34, part 4, "Geospatial Theory, Processing and Applications," Ottawa. See http://astrogeology.usgs.gov/Projects/ISPRS/MEETINGS/ottawa/index.html for on-line abstract.

Franz, M. and D. Harper (2002). „Heliospheric Coordinate Systems," Planetary and Space Science, 50, 2, 217-233.

Greeley, R. and R. M. Batson (1990). Planetary Mapping, Cambridge University Press, Cambridge, 296 p.

Herring, T. A., P. M. Mathews, and B. A. Buffett (2002). Modeling of nutation-precession: Very long baseline interferometry results, J. Geophys. Res., 107(B4), 2069, doi:10.1029/2001JB000165.

International Astronomical Union (2002). Proceedings of the Twenty-Fourth General Assembly, Manchester 2000, Transactions of the IAU, Vol. XXIV-B, pp. 33-57. See http://syrte.obspm.fr/IAU_resolutions/Resol-UAI.htm

Kovalevsky, J. and I. I. Mueller (1981). "Comments on Conventional Terrestrial and Quasi-Inertial Reference Systems," pp. 375-384, in E. M. Gaposchkin and B. Kolaczek, eds., Reference

Coordinate Systems for Earth Dynamics, D. Reidel Publishing Co., Dordrecht, Holland.

Ma, C., E. F. Arias, T. M. Eubanks, A. L. Fey, A.-M. Gontier, C. S. Jacobs, O. J. Sovers, B. A. Archinal, and P. Charlot (1998). "The International Celestial Reference Frame As Realized by Very Long Baseline Interferometry," Astronomical Journal, 116, pp. 516-546, July.

Lunar Geodesy and Cartography Working Group (2008). "Recommendations for Formatting Large Lunar Datasets," May 18, draft. Available at ftp://ftpext.usgs.gov/pub/wr/az/flagstaff/barchinal/LGCWG/MosaickingRecommendations_080930-draft.doc.

Lunar Reconnaissance Orbiter Project (2008). "A Standardized Lunar Coordinate System for the Lunar Reconnaissance Orbiter," Version 4, May 14. Available at http://lunar.gsfc.nasa.gov/library/451-SCI-000958.pdf.

Ma, C., E. F. Arias, T. M. Eubanks, A. L. Fey, A.-M. Gontier, C. S. Jacobs, O. J. Sovers, B. A. Archinal, and P. Charlot (1998). "The International Celestial Reference Frame As Realized by Very Long Baseline Interferometry," Astronomical Journal, 116, pp. 516-546.

Nautical Almanac Office, U. S. Naval Observatory and H. M. Nautical Almanac Office, Royal Greenwich Observatory (1983). "The Introduction of the Improved IAU System of Astronomical Constants, Time Scales and Reference Frame Into the Astronomical Almanac," Supplement to the Astronomical Almanac 1984 (U. S. Government Printing Office, Washington; Her Majesty's Stationary Office, London).

Planetary Cartography Working Group (1993). "Planetary Cartography 1993-2003," NASA, Washington, D.C.

Planetary Data System (2008). Planetary Science Data Dictionary Document, JPL D-7116, Rev. F, October 20. Available at http://pds.nasa.gov/documents/psdd/PSDDmain_1r71.pdf. Also see the PDS Data Dictionary Lookup tool at http://pds.nasa.gov/tools/data_dictionary_lookup.cfm for current keyword definitions.

Russell, C. T. (1971). "Geophysical Coordinate Transformations," Cosmic Electrodynamics, 2, 184-196.

Seidelmann, P. K. (Chair), V. K. Abalakin, M. Bursa, M. E. Davies, C. De Bergh, J. H. Lieske, J. Oberst, J. L. Simon, E. M Standish, P. Stooke, and P. C. Thomas (2002). "Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotation Elements of the Planets and Satellites: 2000," Celestial Mechanics and Dynamical Astronomy, 82, pp. 83-110. [WGCCRE #8]

Seidelmann, P. K. (Chair), B. A. Archinal (Vice-Chair), M. F. A'Hearn, D. P. Cruikshank, J. L. Hilton, H. U. Keller, J. Oberst, J. L. Simon, P. Stooke, D. J. Tholen, and P. C. Thomas (2005), "Report Of The IAU/IAG Working Group On Cartographic Coordinates And Rotational Elements: 2003," Celestial Mechanics and Dynamical Astronomy, 91, pp. 203-215. [WGCCRE

#9]

Seidelmann, P. K. (Chair), B. A. Archinal (Vice-Chair), M. F. A'Hearn, A. Conrad, G. J. Consolmagno, D. Hestroffer, J. L. Hilton, G. A. Krasinsky, G. Neumann, J. Oberst, P. Stooke, E. Tedesco, D. J. Tholen, P. C. Thomas, and I. P. Williams (2007). "Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements: 2006," Celestial Mechanics and Dynamical Astronomy, 98, 155-180. [WGCCRE #10]

Seidelmann, P. K. B. Guinot, and L. E. Dogget, 1992, "Time", Chapter 2, Explanatory Supplement to the Astronomical Almanac, Seidelmann, P. K., ed., U. S. Naval Observatory, University Science Books, Mill Valley, CA.

(This page intentionally left blank.)

Chapter 3. DATA_TYPE Values and Data File Storage Formats

Each PDS archived product is described using label objects that provide information about the data types of stored values. The data elements DATA_TYPE, BIT_DATA_TYPE, and SAMPLE_TYPE appear together with related elements defining starting location and length for each field. In PDS data object definitions the byte, bit, and record positions are counted from left to right, or first to last encountered, and always begin with 1.

Data files may be in ASCII or binary format. ASCII format is often more easily transferred between hardware systems or even application programs on the same computer. Notwithstanding, numeric data are often stored in binary files when the ASCII representation would require substantially more storage space. (For example, each 8-bit signed pixel value in a binary image file would require a four-byte field if stored as an ASCII table.)

3.1 Data Elements

Table 3.1 identifies by object the data elements providing type, location, and length information. The elements ITEMS and ITEM_BYTES are used to subdivide a single COLUMN, FIELD, BIT_COLUMN, or HISTOGRAM into a regular vector containing as many elements as specified for the value of ITEMS. In these objects the DATA_TYPE must indicate the type of a single item in the vector. In the past, the data element ITEM_TYPE was used for this purpose, but DATA_TYPE is now the preferred parameter.

3.2 Data Types

Table 3.2 identifies the valid values for the DATA_TYPE, BIT_DATA_TYPE, and SAMPLE_TYPE data elements used in PDS data object definitions. The values for these elements must be one of the standard values listed in the *Planetary Science Data Dictionary* (PSDD). Please note:

- In all cases, these standard values refer to the physical storage format of the data in the data file.
- In some cases, obsolete values from previous versions of the PDS Standards have been retained as aliases for more specific values (the type "INTEGER", for example, is interpreted as "MSB_INTEGER" when it is encountered). In these cases the more specific value should always be used in new data sets – the obsolete value is retained only for backward compatibility. Obsolete values are indicated in the table.
- Aliases have been supplied for some of the generic data types that indicate the kind of system on which the data originated. For example, "MAC_REAL" is an alias for "IEEE_REAL", but "VAX_REAL" has no alias, as the VAX binary storage format is unique to VAX systems. In general, the more generic term is preferred, but the system-specific version may be used if needed.

Table 3.1: Type Elements Used in Data Label Objects

Data Object	**Data Elements**	**Notes**
COLUMN	DATA_TYPE	
(without ITEMS)	START_BYTE	
	BYTES	
COLUMN	DATA_TYPE	alias for ITEM_TYPE
(with ITEMS)	START_BYTE	
	BYTES *(optional)*	total bytes in COLUMN
	ITEMS	
	ITEM_BYTES	bytes in each ITEM
BIT_COLUMN	BIT_DATA_TYPE	
(without ITEMS)	START_BIT	
	BITS	
BIT_COLUMN	START_BIT	
(with ITEMS)	BITS (*optional*)	total bits in BIT_COLUMN
	ITEMS	
	ITEM_BITS	bits in each ITEM
FIELD	DATA_TYPE	if populated
(no items)	FIELD_NUMBER	
	BYTES	maximum FIELD bytes
FIELD	DATA_TYPE	if populated
(with items)	FIELD_NUMBER	
	BYTES	maximum bytes in FIELD
	ITEMS	
	ITEM_BYTES	maximum item bytes
IMAGE	SAMPLE_TYPE	
	SAMPLE_BITS	
HISTOGRAM	DATA_TYPE	alias for ITEM_TYPE
	BYTES (*optional*)	total bytes in HISTOGRAM
	ITEMS	number of bins in HISTOGRAM
	ITEM_BYTES	bytes in each ITEM

Table 3.2: Standard PDS Data Types

Data Element Usage Codes:

D = DATA_TYPE
B = BIT_DATA_TYPE
S = SAMPLE_TYPE

Usage	Value	Description
D	ASCII_REAL	ASCII character string representing a real number; see Section 5.4 for formatting rules
D	ASCII_INTEGER	ASCII character string representing an integer; see Section 5.4 for formatting rules
D	ASCII_COMPLEX	ASCII character string representing a complex number; see Section 5.4 for formatting rules
Obsolete	BIT_STRING	alias for MSB_BIT_STRING
D, B	BOOLEAN	True/False Indicator: a 1-, 2- or 4-byte integer or 1-32 bit number. All 0 = False; anything else = True.
D	CHARACTER	ASCII character string; see Section 5.4 for formatting rules
Obsolete	COMPLEX	alias for IEEE_COMPLEX
D	DATE	ASCII character string representing a date in PDS standard format; see Section 5.4 for formatting rules
D	EBCDIC_CHARACTER	EBCDIC character string
Obsolete	FLOAT	alias for IEEE_REAL
D	IBM_COMPLEX	IBM 360/370 mainframe complex number (8- or 16-byte)
D, S	IBM_INTEGER	IBM 360/370 mainframe 1-, 2-, and 4-byte signed integers
D, S	IBM_REAL	IBM 360/370 mainframe real number (4- or 8-byte)
D, B, S	IBM_UNSIGNED_INTEGER	IBM 360/370 mainframe 1-, 2-, and 4-byte unsigned integers
D	IEEE_COMPLEX	8-, 16-, and 20-byte complex numbers
D, S	IEEE_REAL	4-, 8- and 10-byte real numbers
Obsolete	INTEGER	alias for MSB_INTEGER
D	LSB_BIT_STRING	1-, 2-, and 4-byte bit strings
D, S	LSB_INTEGER	1-, 2-, and 4-byte signed integers
D, B, S	LSB_UNSIGNED_INTEGER	1-, 2-, and 4-byte unsigned integers
D	MAC_COMPLEX	alias for IEEE_COMPLEX
D, S	MAC_INTEGER	alias for MSB_INTEGER
D, S	MAC_REAL	alias for IEEE_REAL
D, B, S	MAC_UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	MSB_BIT_STRING	1-, 2-, and 4-byte bit strings
D, S	MSB_INTEGER	1-, 2-, and 4-byte signed integers
D, B, S	MSB_UNSIGNED_INTEGER	1-, 2-, and 4-byte unsigned integers
D, B	N/A	Used only for spare (or unused) fields included in the data file.

D	PC_COMPLEX	8-, 16-, and 20-byte complex numbers in IBM/PC format
D, S	PC_INTEGER	alias for LSB_INTEGER
D, S	PC_REAL	4-, 8-, and 10-byte real numbers in IBM/PC format
D, B, S	PC_UNSIGNED_INTEGER	alias for LSB_UNSIGNED_INTEGER
Obsolete	REAL	alias for IEEE_REAL
D	SUN_COMPLEX	alias for IEEE_COMPLEX
D, S	SUN_INTEGER	alias for MSB_INTEGER
D, S	SUN_REAL	alias for IEEE_REAL
D, B, S	SUN_UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	TIME	ASCII character string representing a date/time in PDS standard format; see Section 5.4 for formatting rules
Obsolete	UNSIGNED_INTEGER	alias for MSB_UNSIGNED_INTEGER
D	VAX_BIT_STRING	alias for LSB_BIT_STRING
D	VAX_COMPLEX	Vax F-, D-, and H-type (8-, 16- and 32-byte, respectively) complex numbers
D, S	VAX_DOUBLE	alias for VAX_REAL
D, S	VAX_INTEGER	alias for LSB_INTEGER
D, S	VAX_REAL	Vax F-, D-, and H-type (4-, 8- and 16-byte, respectively) real numbers
D, B, S	VAX_UNSIGNED_INTEGER	alias for LSB_UNSIGNED_INTEGER
D	VAXG_COMPLEX	Vax G-type (16-byte) complex numbers
D, S	VAXG_REAL	Vax G-type (8-byte) real numbers

3.3 Binary Integers

There are two widely used formats for integer representations in 16-bit and 32-bit binary fields: most significant byte first (MSB) and least significant byte first (LSB) architectures. The MSB architectures include IBM mainframes, many UNIX systems such as SUN, and Macintosh computers. The LSB architectures include VAX systems and IBM PCs. In the original PDS system the default format was MSB, thus the designation of "INTEGER" and "UNSIGNED_INTEGER" as aliases of "MSB_INTEGER" and "MSB_UNSIGNED_INTEGER". New data sets should be prepared using the appropriate specific designation from Table 3.2, above.

3.4 Signed vs. Unsigned Integers

The "_INTEGER" data types refer to signed, 2's complement integers. Use the corresponding "_UNSIGNED_INTEGER" type for unsigned integer and bit string fields.

3.5 Floating Point Formats

The PDS default representation for floating point numbers is the ANSI/IEEE standard. This representation is defined as the IEEE_REAL data type, with aliases identified in Table 3.2. Several additional specific floating-point representations supported by PDS are described in Appendix C.

3.6 Bit String Data

The BIT_STRING data types are used in definitions of table columns holding individual bit field values. A BIT_COLUMN object defines each bit field. BIT_STRING data types can be 1-, 2-, or 4-byte fields, much like a binary integer. Extraction of specific bit fields within a 2- or 4-byte BIT_STRING is dependent on the host architecture (MSB or LSB). In interpreting bit fields (BIT_COLUMNS) within a BIT_STRING, any necessary conversions such as byte swapping from LSB to MSB are done first, then bit field values (START_BIT, BITS) are used to extract the appropriate bits. This procedure ensures that bit fields are not fragmented due to differences in hardware architectures.

3.7 Character Data

Specification of character field format in ASCII and binary files pending.

3.8 Format Specifications

Data format specifications provided in the FORMAT element serve two purposes:

1. In an ASCII TABLE data file or SPREADSHEET file, they provide a format which can be used in scanning the ASCII record for individual fields; and
2. In a binary data file, they provide a format that can be used to display the data values.

A subset of the FORTRAN data format specifiers is used for the values of FORMAT elements. Valid specifiers include:

Aw	Character data value
Iw	Integer value
Fw.d	Floating point value, displayed in decimal format
Ew.d[Ee]	Floating point value, displayed in exponential format

Where:

w is the total number of positions in the output field (including sign, decimal point, and exponentiation character – usually “E” – if any);
d is the number of positions to the right of the decimal point;
e is the number of positions in exponent length field.

3.9 Internal Representations of Data Types

Appendix C contains the detailed internal representations of the PDS standard data types listed in Table 3.2.

The PDS has developed tools designed to use the specifications contained in Appendix C for interpreting data values for display and validation.

Chapter 4. Data Objects and Products

At its simplest, a *data product* consists of a PDS label and the data object that it describes. More complex data products may contain several mutually dependent data objects, a primary object and one or more secondary objects, or both. In all cases, a single label is used to describe all parts of the product (even if they are held in separate physical files). A single PRODUCT_ID value is defined for the entire set in that PDS label.

A data product is one component of a *data set* (see the *Data Set/Data Set Collection Contents and Naming* chapter of this document).

Primary Data Object

A primary data object is a set of results from a scientific observation. Primary data objects are usually described using one of these PDS object structures:

TABLE
SPREADSHEET
IMAGE
SERIES
SPECTRUM
QUBE

Secondary Data Object

A secondary data object is any data used for processing or interpreting the primary data object(s), for example, a histogram derived from an image. Secondary data objects are usually described using one of these PDS object structures:

HISTOGRAM
PALETTE
HEADER

The PDS data product label, written in Object Description Language (ODL) (see the *Object Description Language (ODL) Specification and Usage* chapter of this document), defines both the physical and logical structure of the constituent data object(s).

4.1 Data Product File Configurations

The PDS label and data object may be in the same file or separate files. For data products with more than one object, the data objects may be in one or more files. In all cases, however, there must be exactly one PDS label containing exactly one PRODUCT_ID value. The PRODUCT_ID value must be unique within the data set containing this data product.

Example

Consider a data product that consists of a 3-color image in which each color plane is stored in a separate physical file (that is, one file each for red, blue and green). Since all three colors are required to get the full image, this product contains three mutually dependent primary objects.

The label for this data product will contain a single PRODUCT_ID, three pointers to the separate data files, and three IMAGE object definitions. To aid in distinguishing between data files, the data preparer may also choose to include an IMAGE_ID keyword in each IMAGE object definition. The resulting PDS label would contain the following lines:

```
PRODUCT_ID          = "22A190"
…
^RED_IMAGE          = "22A190R.IMG"
^GREEN_IMAGE        = "22A190G.IMG"
^BLUE_IMAGE         = "22A190B.IMG"
…
OBJECT              = RED_IMAGE
   IMAGE_ID         = "22A190-RED"
   …
END_OBJECT          = RED_IMAGE

OBJECT              = GREEN_IMAGE
   IMAGE_ID         = "22A190-GREEN"
   …
END_OBJECT          = GREEN_IMAGE

OBJECT              = BLUE_IMAGE
   IMAGE_ID         = "22A190-BLUE"
   …
END_OBJECT          = BLUE_IMAGE
```

Figure 4.1 illustrates file configurations for a data product with a single data object.

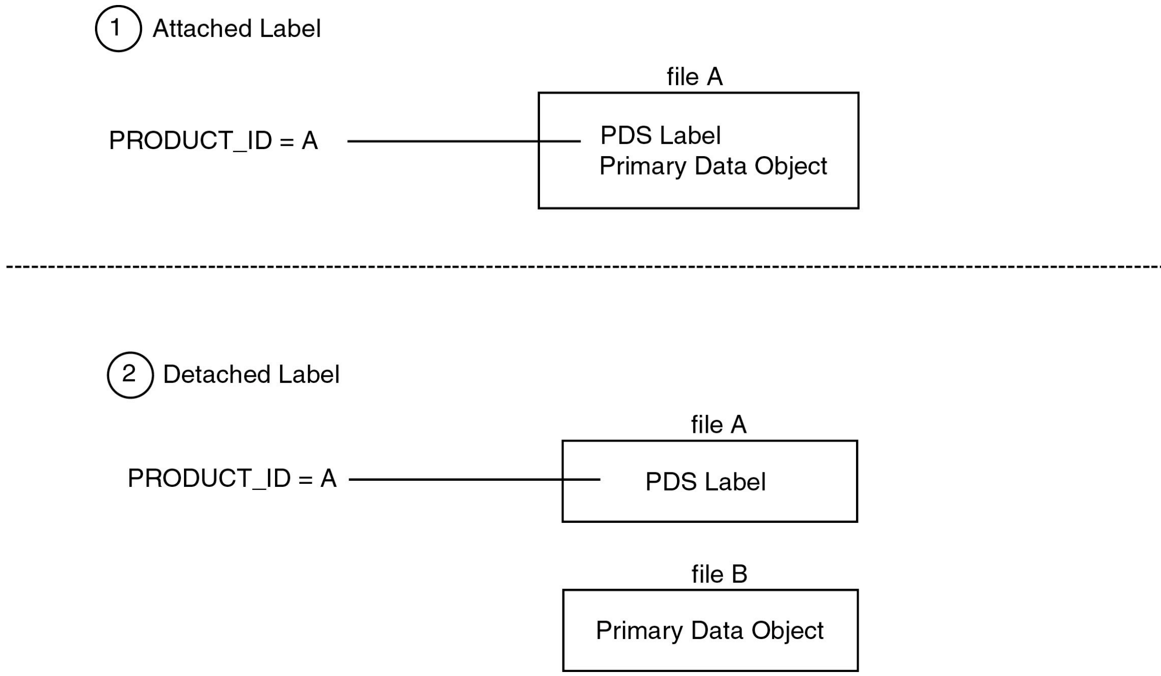


Figure 4.1 Data Product with a Single Data Object

Figure 4.2 shows the possible file configurations for a single data product consisting of one primary and one secondary data object. Similar examples could be made using data products composed of more than two data objects.

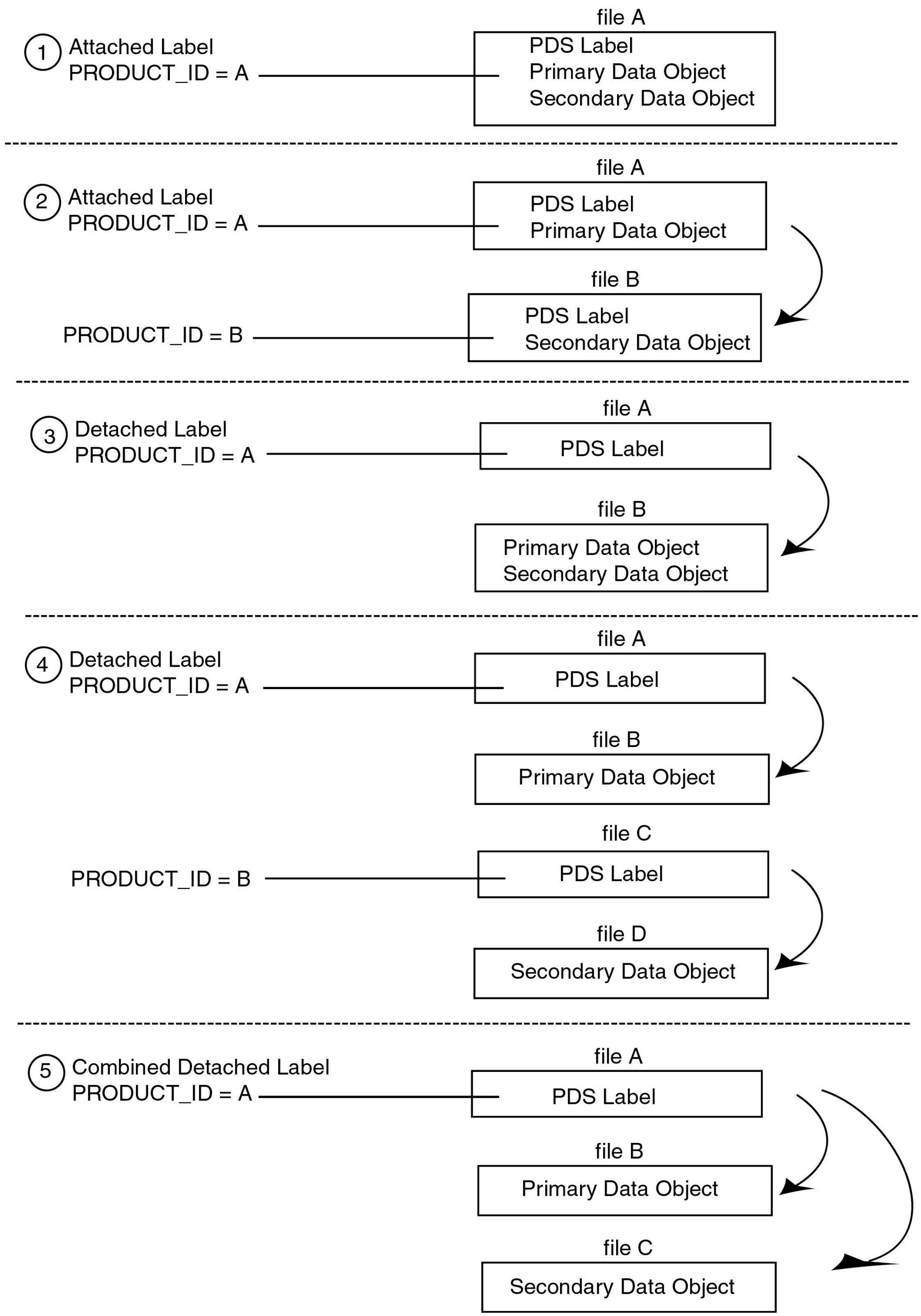


Figure 4-2. Data Product with Multiple Data Objects

Chapter 5. Data Product Labels

PDS data product labels are required for describing the contents and format of each individual data product within a data set. PDS data product labels are written in the Object Description Language (ODL). The PDS has chosen to label the wide variety of data products under archival preparation by implementing a standard set of data object definitions, group definitions, data elements, and standard values for the elements. These data object definitions, data elements, and standard values are defined in the *Planetary Science Data Dictionary* (PSDD). Appendix A of this document provides general descriptions and examples of the use of these data object definitions and data elements for labeling data products.

5.1 Format of PDS Labels

5.1.1 Labeling methods

In order to identify and describe the organization, content, and format of each data product, PDS requires a distinct data product label for each individual data product file. These distinct product labels may be constructed in one of three ways:

<u>Attached</u> - The PDS data product label is attached at the beginning of the data product file. There is one label attached to each data product file.

<u>Detached</u> - The PDS data product label is detached from the data and resides in a separate file which contains a pointer to the data product file. There is one detached label file for every data product file. The label file should have the same base name as its associated data file, but the extension .LBL .

<u>Combined Detached</u> - A single PDS detached data product label file is used to describe the contents of more than one data product file. The combined detached label contains pointers to individual data products.

NOTE: Although all three labeling methods are equally acceptable, the PDS tools do not currently support the Combined Detached label option.

Figure 5.1 illustrates the use of each of these methods for labeling individual data product files.

5.1.2 Label format

PDS recommends that labels have stream record format, and line lengths of at most 80 characters (including the CR/LF line terminators) so that the entire label can be seen on a computer screen without horizontal scrolling. The carriage return and line feed (CR/LF) pair is the required line terminator for all PDS labels. (See the *Record Formats* chapter of this document.)

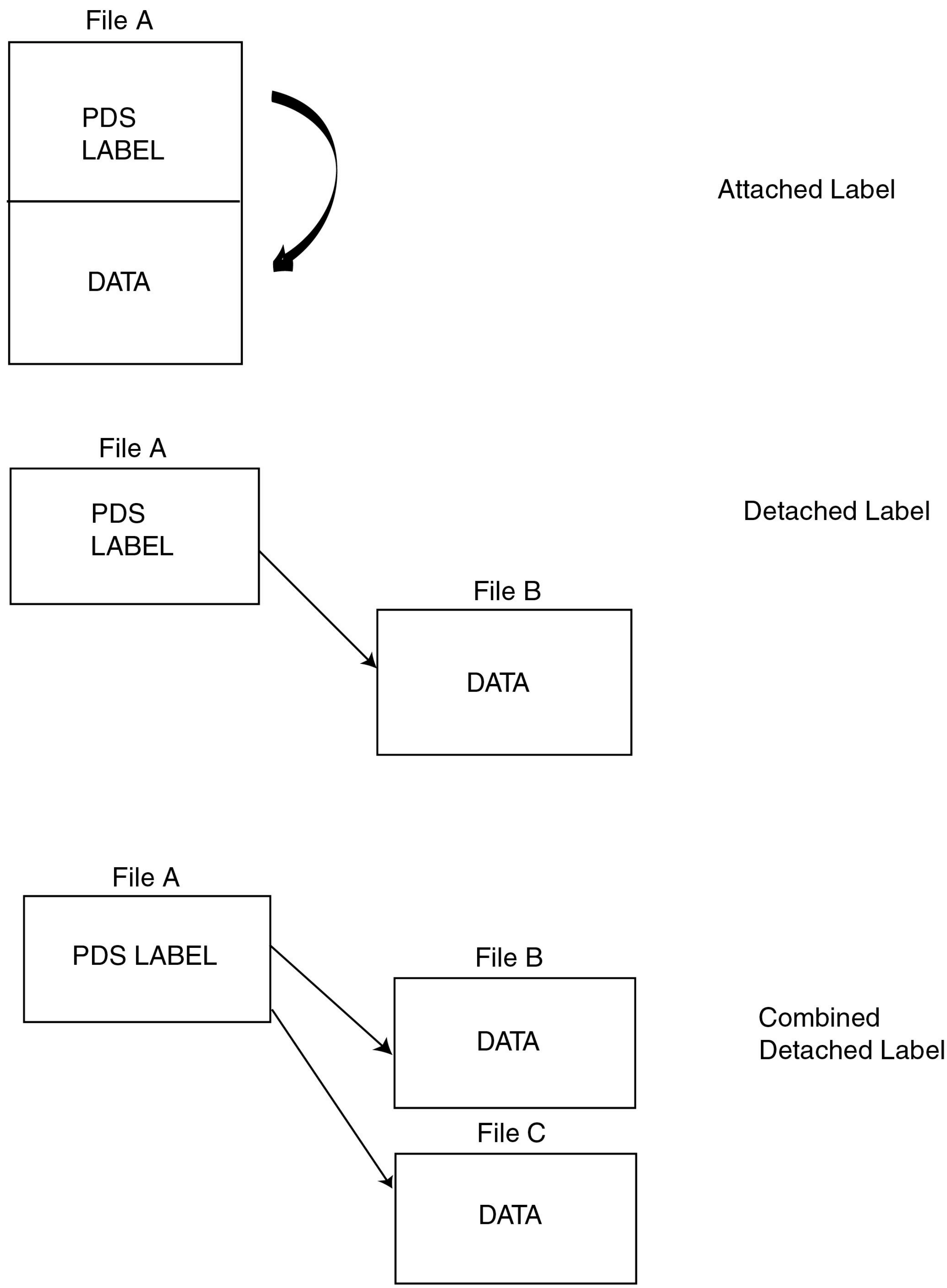


Figure 5.1 Attached, Detached, and Combined Detached PDS Labels

All values in a PDS label should be in upper case, except values for descriptive elements (DESCRIPTION, NOTE, etc.). It is also recommended that the equal signs in the labels be aligned for ease of reading.

ASCII Character Set

All values in a PDS label must conform to the standard 7-bit ASCII character set. Labels may include characters in the range of ASCII characters 32 through 127 (decimal), and the record delimiters Line Feed (10 decimal) and Carriage Return (13 decimal).

The remaining 7-bit ASCII characters (1-9, 11, 12, and 14-31 decimal, which includes the horizontal and vertical tab and form feed characters) are not permitted in PDS labels. Note that the 8-bit characters 128 through 255 (decimal) are not used in the PDS as the interpretation of these characters varies by operating system, computer platform, and font selected. Specifically, extended-set characters with diacritical marks are not to be used as they are interpreted differently by different applications.

Label Padding

When a fixed length data file has an attached label, the label is padded with space characters (ASCII 32 decimal) in one of the following ways:

1) Spaces are added after the label's END <CR><LF> statement and before the data so that the total of the label (in bytes) is an integral multiple of the record length of the data. In this case, LABEL_RECORDS is calculated by dividing the total padded length of the label section, in bytes, by the stated value of RECORD_BYTES.

Example

In the example below, the label portion of the file is 7 x 324 = 2268 bytes in length, including blank fill between the END<CR><LF> statement and the first byte of data. The actual data portion of the file starts at record 8 (i.e., the 1st byte of the 8th record starts at byte (7 x 324)+1 = 2269)

```
RECORD_TYPE                    = FIXED_LENGTH<CR><LF>
RECORD_BYTES                   = 324<CR><LF>
FILE_RECORDS                   = 334<CR><LF>
LABEL_RECORDS                  = 7<CR><LF>

^IMAGE                         = 8<CR><LF>

END<CR><LF>
....blank fill....
data
```

2) Each line in the label may be padded with space characters so that each line in the label has the same record length as the data file. In this case, the label line length may exceed the recommended 80 characters; LABEL_RECORDS is the number of physical records in the label section of the file.

Example

In the example below, the label portion of the file is 80 x 85 = 6800 bytes in length. Each line in the label portion of the file is 85 bytes long, the same length as each data record. Notice the blank space between the actual values in the label and the line delimiters. In the example, the label is 80 lines long (i.e., 80 records long) and the data begin at record 81. Note that the label is padded so that <CR><LF> are in bytes 84 and 85.

```
RECORD_TYPE              = FIXED_LENGTH    <CR><LF>
RECORD_BYTES             = 85              <CR><LF>
FILE_RECORDS             = 300             <CR><LF>
LABEL_RECORDS            = 80              <CR><LF>
...
^TABLE                   = 81              <CR><LF>
END                                        <CR><LF>
Data
```

5.2 Data Product Label Content

5.2.1 Attached and Detached Labels

PDS data product labels have a general structure that is used for all attached and detached labels, except for data products described by minimal labels. (Minimal labels are described in Section 5.2.3.)

- LABEL STANDARDS identifier
- FILE CHARACTERISTIC data elements
- DATA OBJECT pointers
- IDENTIFICATION data elements
- DESCRIPTIVE data elements
- DATA OBJECT DEFINITIONS
- END statement

Figure 5.2 provides an example of how this general structure appears in an attached or detached label for a data product file containing multiple data objects.

5.2.2 Combined Detached Labels

For the Combined Detached label option, the general label structure is modified slightly to reference each individual file within its own FILE object explicitly. In addition, identification and descriptive data elements that apply to all of the files can be located before the FILE objects.

PDS LABEL

PDS LABEL	
PDS_VERSION_ID = DD_VERSION_ID = LABEL_REVISION_NOTE =	• LABEL STANDARDS IDENTIFIERS
/* FILE_CHARACTERISTICS */ RECORD_TYPE = RECORD_BYTES = FILE_RECORDS = LABEL_RECORDS =	• FILE CHARACTERISTICS DATA ELEMENTS
/* POINTERS TO DATA OBJECTS */ ^IMAGE = ^HISTOGRAM =	• DATA OBJECT POINTERS (primary, secondary)
/* IDENTIFICATION DATA ELEMENTS */ DATA_SET_ID = PRODUCT_ID = SPACECRAFT_NAME = INSTRUMENT_NAME = TARGET_NAME = START_TIME = STOP_TIME = . . . PRODUCT_CREATION_TIME =	• IDENTIFICATION DATA ELEMENTS
/* DESCRIPTIVE DATA ELEMENTS */ FILTER_NAME = OFFSET_MODE_ID = . . .	• DESCRIPTIVE DATA ELEMENTS
/* DATA OBJECT DEFINITIONS */ OBJECT = IMAGE . . . END_OBJECT = IMAGE OBJECT = HISTOGRAM . . . END_OBJECT = HISTOGRAM	• DATA OBJECT DEFINITIONS (primary, secondary)
END	• END STATEMENT

Figure 5.2 PDS Attached / Detached Label Structure

- LABEL STANDARDS identifiers
- IDENTIFICATION data elements that apply to all referenced data files
- DESCRIPTIVE data elements that apply to all referenced data files
- OBJECT=FILE statement (Repeats for each data product file)
    - FILE CHARACTERISTIC data elements
    - DATA OBJECT pointers
    - IDENTIFICATION data elements
    - DESCRIPTIVE data elements
    - DATA OBJECT DEFINITION
- END_OBJECT=FILE statement
- END statement

Figure 5.3 provides an example of how this general structure appears in a combined detached label that describes more than one data product file.

5.2.3 Minimal Labels

Use of the minimal label option is only allowed when the format of the data cannot be supported by any PDS data object structure other than the FILE object.

For minimal labels the required use of data objects is waived. A minimal label does not require any explicit PDS data object definitions or pointers to data objects. This applies to both attached and detached labels.

Minimal labels must satisfy the following requirements:

(1) Provide the ability to locate the data associated with the label.

1a. Attached labels

Since data objects and pointers are not required in the minimal label, by definition the data follow immediately after the label.

1b. Detached Labels

Both the implicit and explicit use of the FILE object are supported. The FILE_NAME keyword is required in the explicit FILE object, or in the label itself if no FILE object is included.

(2) Provide the ability to locate a description of the format/content of the data. One of the following must be provided in the minimal label:

2a. ^DESCRIPTION = “<filename>”
This is a pointer to a file containing a detailed description of the data format, which may be located in the same directory as the data or in the DOCUMENT subdirectory.

PDS LABEL

PDS LABEL	
PDS_VERSION_ID = DD_VERSION_ID = LABEL_REVISION_NOTE =	• LABEL STANDARDS IDENTIFIERS
DATA_SET_ID = PRODUCT_ID = SPACECRAFT_ID = INSTRUMENT_NAME = TARGET_NAME = PRODUCT_CREATION_TIME =	• IDENTIFICATION & DESCRIPTIVE DATA ELEMENTS for all files
OBJECT = FILE RECORD_TYPE = . . . FILE_RECORDS = ^TIME_SERIES = "FILEA" START_TIME = STOP_TIME = OBJECT = TIME_SERIES . . . END_OBJECT = TIME_SERIES END_OBJECT = FILE	For detached FILE A: • FILE CHARACTERISTICS DATA ELEMENTS • DATA OBJECT POINTERS • IDENTIFICATION / DESCRIPTIVE DATA ELEMENTS • DATA OBJECT DEFINITIONS
OBJECT = FILE RECORD_TYPE = . . . FILE_RECORDS = ^TIME_SERIES = "FILEB" START_TIME = STOP_TIME = OBJECT = TIME_SERIES . . . END_OBJECT = TIME_SERIES END_OBJECT = FILE	For detached FILE B: • FILE CHARACTERISTICS DATA ELEMENTS • DATA OBJECT POINTERS • IDENTIFICATION / DESCRIPTIVE DATA ELEMENTS • DATA OBJECT DEFINITIONS
END	• END STATEMENT

Figure 5.3 PDS Combined / Detached PDS Label Structure

2b. DESCRIPTION = “<text appears here>”
This is either a detailed description of the data file, its format, data types, and use, or it is a reference to a document available externally, e.g., a Software Interface Specification (SIS) or similar document.

(3) When minimal labels are used, DATA_OBJECT_TYPE = FILE should be used in the DATA_SET catalog file

5.2.3.1 Implicit File Object (Attached and Detached Minimal Label)

The general structure for minimal labels with implicit file objects is as follows:

- LABEL STANDARDS identifiers
- FILE CHARACTERISTIC data elements
- IDENTIFICATION data elements
- DESCRIPTIVE data elements
- END statement

5.2.3.2 Explicit File Object (Detached Minimal Label)

The general structure for minimal labels with explicit file objects is as follows:

- LABEL STANDARDS identifiers
- IDENTIFICATION data elements
- DESCRIPTIVE data elements
- OBJECT=FILE statement
    - FILE CHARACTERISTIC data elements

- END_OBJECT=FILE
- END statements

Figure 5.4 provides an example of how this general structure appears in a detached minimal label. In this example, an implicit FILE object is used.

5.3 Detailed Label Contents Description

This section describes the detailed requirements for the content of PDS labels. The subsections describe label standards identifiers, file characteristic data elements, data object pointers, identification data elements, descriptive data elements, data object definitions, and the END statement. Note that identifiers, pointers and data elements may only appear once within a given object definition (including an implicit object definition, such as a minimal label). If multiple values are needed, they must be represented using either the sequence or set syntax (see Sections 12.5.5 and 12.5.6, respectively).

PDS LABEL

PDS LABEL	
PDS_VERSION_ID = DD_VERSION_ID = LABEL_REVISION_NOTE =	• LABEL STANDARDS IDENTIFIERS
/* FILE_CHARACTERISTICS */ RECORD_TYPE = RECORD_BYTES = FILE_NAME = FILE_RECORDS = LABEL_RECORDS =	• FILE CHARACTERISTICS DATA ELEMENTS
/* IDENTIFICATION DATA ELEMENTS */ DATA_SET_ID = PRODUCT_ID = SPACECRAFT_NAME = INSTRUMENT_NAME = TARGET_NAME = START_TIME = STOP_TIME = . . . PRODUCT_CREATION_TIME =	• IDENTIFICATION DATA ELEMENTS
/* DESCRIPTIVE DATA ELEMENTS */ FILTER_NAME = OFFSET_MODE_ID = ^DESCRIPTION = . . .	• DESCRIPTIVE DATA ELEMENTS
END	• END STATEMENT

Figure 5.4 PDS Detached Minimal Label Structure

5.3.1 Label Standards Identifiers

Each PDS label must begin with the PDS_VERSION_ID data element. This element identifies the published version of the Standards to which the label adheres, for purposes of both validation as well as software development and support. For labels adhering to the standards described in this document (the *PDS Standards Reference*, Version 3.4), the appropriate value is “PDS3”:

```
PDS_VERSION_ID = PDS3
```

The PDS does not require Standard Formatted Data Unit (SFDU) labels on individual products, but they may be desired for conformance with specific project or other agency requirements. When SFDU labels are provided on a PDS data product, the SFDU label must *precede* the PDS_VERSION_ID keyword, thus:

```
CCSD....                        [optional SFDU label]
PDS_VERSION_ID
DD_VERSION_ID
LABEL_REVISION_NOTE
```

SFDU labels in PDS products must follow the format standards described in *SFDU Usage* chapter in this document.

The DD_VERSION_ID element identifies the version of the PDS Data Dictionary to which a label complies. Current PDS practice is to identify a Data Dictionary version with the identifier used for the PDS catalog build in which it resides, e.g., pdscat1r47, pdscat1r48, and so on. This keyword will use the upper case representation of the catalog identifier, e.g., PDSCAT1R47, PDSCAT1R48, etc.

The LABEL_REVISION_NOTE element is a free form, unlimited-length character string providing information regarding the revision status and authorship of a PDS label. It should include at least the latest revision date and the author of the current version, but may include a complete editing history. This element is required in all catalog labels.

Example

```
PDS_VERSION_ID                  = PDS3
DD_VERSION_ID                  = PDSCAT1R52
LABEL_REVISION_NOTE             = "1999-08-01, Anne Raugh (SBN), initial
release;"
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 80
```

5.3.2 File Characteristic Data Elements

PDS data product labels contain data element information that describes important attributes of the physical structure of a data product file. The PDS file characteristic data elements are:

RECORD_TYPE
RECORD_BYTES
FILE_RECORDS
LABEL_RECORDS

The RECORD_TYPE data element identifies the record characteristics of the data product file. A complete discussion of the RECORD_TYPE data element and its use in describing data products produced on various platforms is provided in the *Record Formats* chapter in this document. The RECORD_BYTES data element identifies the number of bytes in each physical record in the data product file. The FILE_RECORDS data element identifies the number of physical records in the file. The LABEL_RECORDS data element identifies the number of physical records that make up the PDS product label.

Not all of these data elements are required in every data product label. Table 5.1 lists the required (Req) and optional (Opt) file characteristic data elements for a variety of data products and labeling methods for both attached (Att) and detached (Det) labels. Where (max) is specified, the value indicates the maximum size of any physical record in the file.

Table 5.1: File Characteristic Data Element Requirements

Labeling Method	Att	Det	Att	Det	Att	Det	Att	Det
RECORD_TYPE	FIXED_LENGTH		VARIABLE_LENGTH		STREAM		UNDEFINED	
RECORD_BYTES	Req	Req	Rmax	Rmax	Omax	-	-	-
FILE_RECORDS	Req	Req	Req	Req	Opt	Opt	-	-
LABEL_RECORDS	Req	-	Req	-	Opt	-	-	-

Note: The FILE_NAME keyword is required in detached minimal labels.

5.3.3 Data Object Pointers

“Data objects” are the actual data for which the structure and attributes are defined in a PDS label. Each data product file contains one or more data objects. The PDS uses a pointer within the product labels to identify the file locations for all objects in a data product.

Example

```
^TABLE = "DATA.DAT"
^TABLE = ("DATA.DAT", 10 <BYTES>)
```

5.3.3.1 Use of Pointers in Attached Labels

Data object pointers are required in labels with one exception: attached labels that refer to only a single object. In the absence of a pointer, the data object is assumed to start in the next physical record after the PDS product label area. This is commonly the case with ASCII text files described by a TEXT object and ASCII SPICE files described by a SPICE_KERNEL object. The top two illustrations in Figure 5.5 show example files that do not require data object pointers.

Object pointers are required for all data objects, even when multiple data objects are stored in a single data product file. Data object pointers in attached labels take one of two forms:

^<object_identifier> = nnn

where nnn represents the starting record number within the file (first record is numbered 1), Or,

^<object_identifier> = nnn <BYTES>

where nnn represents the starting byte location within the file (first byte is numbered 1).

See Chapter 12, *Object Description Language (ODL) Specification and Usage*, and Chapter 14, *Pointer Usage*, in this document for a complete description of pointer syntax.

The bottom two illustrations in Figure 5.5 show the use of required data object pointers for attached label products containing multiple data objects.

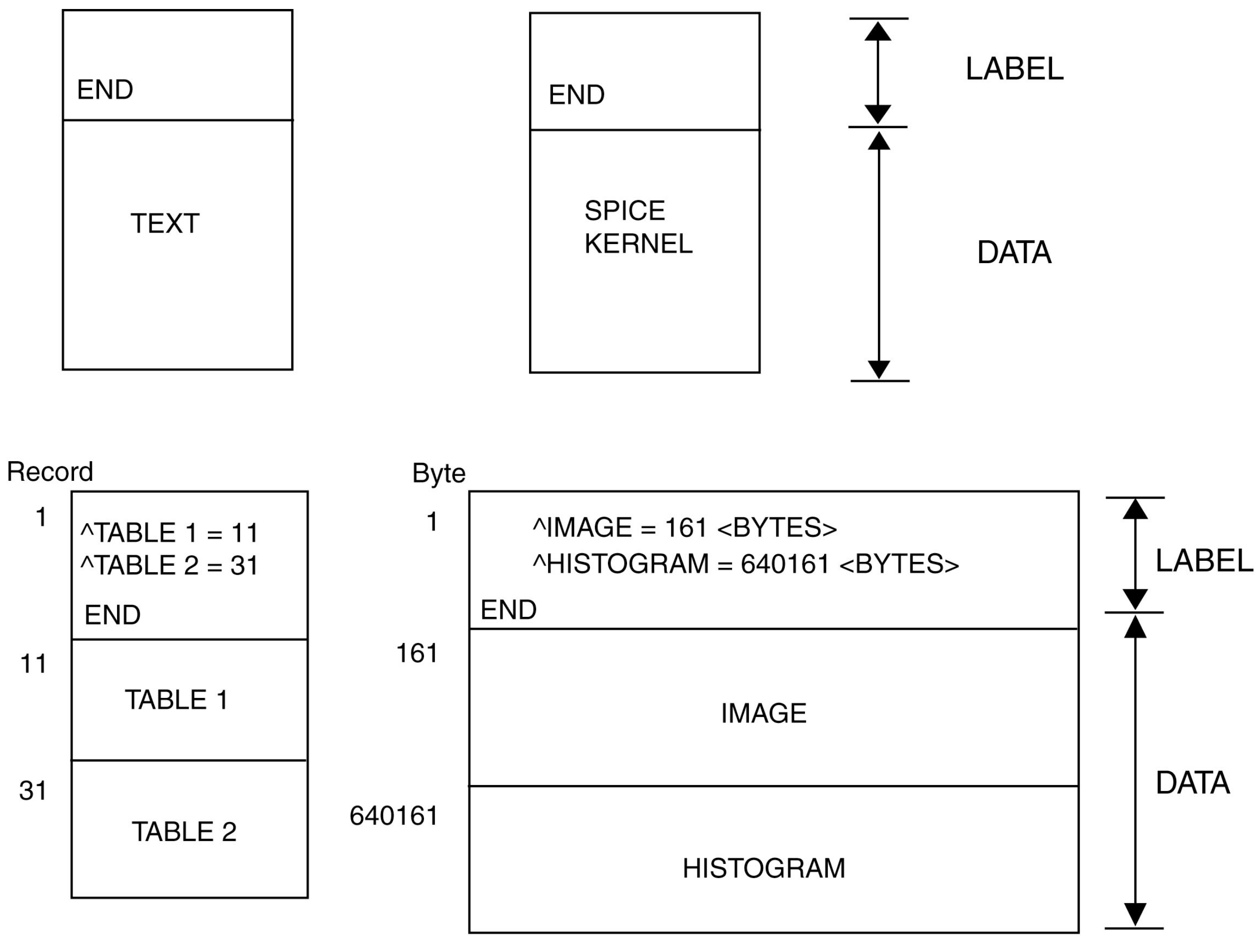


Figure 5.5 Data Object Pointers-Attached Labels

5.3.3.2 Use of Pointers in Detached and Combined Detached Labels

When the PDS data product label is a detached or a combined detached label, data object pointers are required for all data objects referenced.

The syntax for these data object pointers takes one of three forms:

(1) ^*object_identifier* = "filename"
(2) ^*object_identifier* = ("filename", nnn)
(3) ^*object_identifier* = ("filename", nnn <BYTES>)

With respect to the above three cases:

(a) These object pointers reference either byte or record locations in data files that are detached, or separate from, the label file.
(b) "Filename" is the name of the detached data file. File names must be in uppercase characters.
(c) When no offset is specified, the first record is assumed.
(d) Records and bytes are numbered from 1.

In the first case, the data object is located at the beginning of the referenced file. In the second case, the data object begins with the nnn[th] physical record from the beginning of the referenced file. In the third case, the data object begins with the nnn[th] byte from the beginning of the referenced file.

Examples

```
^IMAGE                    = ("DATA.IMG")
^ENGINEERING_TABLE        = ("DATA.DAT", 10)
^TABLE                    = ("DATA.TAB", 10 <BYTES>)
```

Figure 5.6 contains several examples of data object pointer usage for data product files with detached or combined detached labels. The top example shows a data product consisting of a HEADER data object and a TABLE data object together in a single file. The detached label for this product includes pointers for both data objects, with the TABLE object starting at byte 601 of file A. The middle example illustrates a combined detached label for a data product contained in two data objects, each in a separate file. A separate pointer is provided for each data object. The bottom example shows a detached label for a data product containing multiple data objects.

The third example shows a complex data file structure. The HEADER object comes first in the data file and, as the pointer ("^HEADER") shows, it requires no explicit offset (record 1 is assumed). Two parallel objects, a TABLE and an IMAGE, then follow the header. For this section of the file, each record contains one row of the TABLE followed by one line of the IMAGE. In the TABLE object description, the bytes of the IMAGE are accounted for as ROW_SUFFIX_BYTES; in the IMAGE object description, the bytes of the TABLE object are accounted for as LINE_PREFIX_BYTES. Both objects start in the same record, and therefore have the same offset (4). See the IMAGE and TABLE object descriptions for more information on prefix and suffix bytes. Had this data file been organized sequentially (so that, for example, the HEADER was followed by the TABLE, which in turn was followed by the IMAGE), then each object would have had its own offset.

5.3.3.3 Note Concerning Minimal Attached and Detached Labels

Data object pointers do not exist in minimal labels. In these cases the format of the data is usually fully described in a separate file or document.

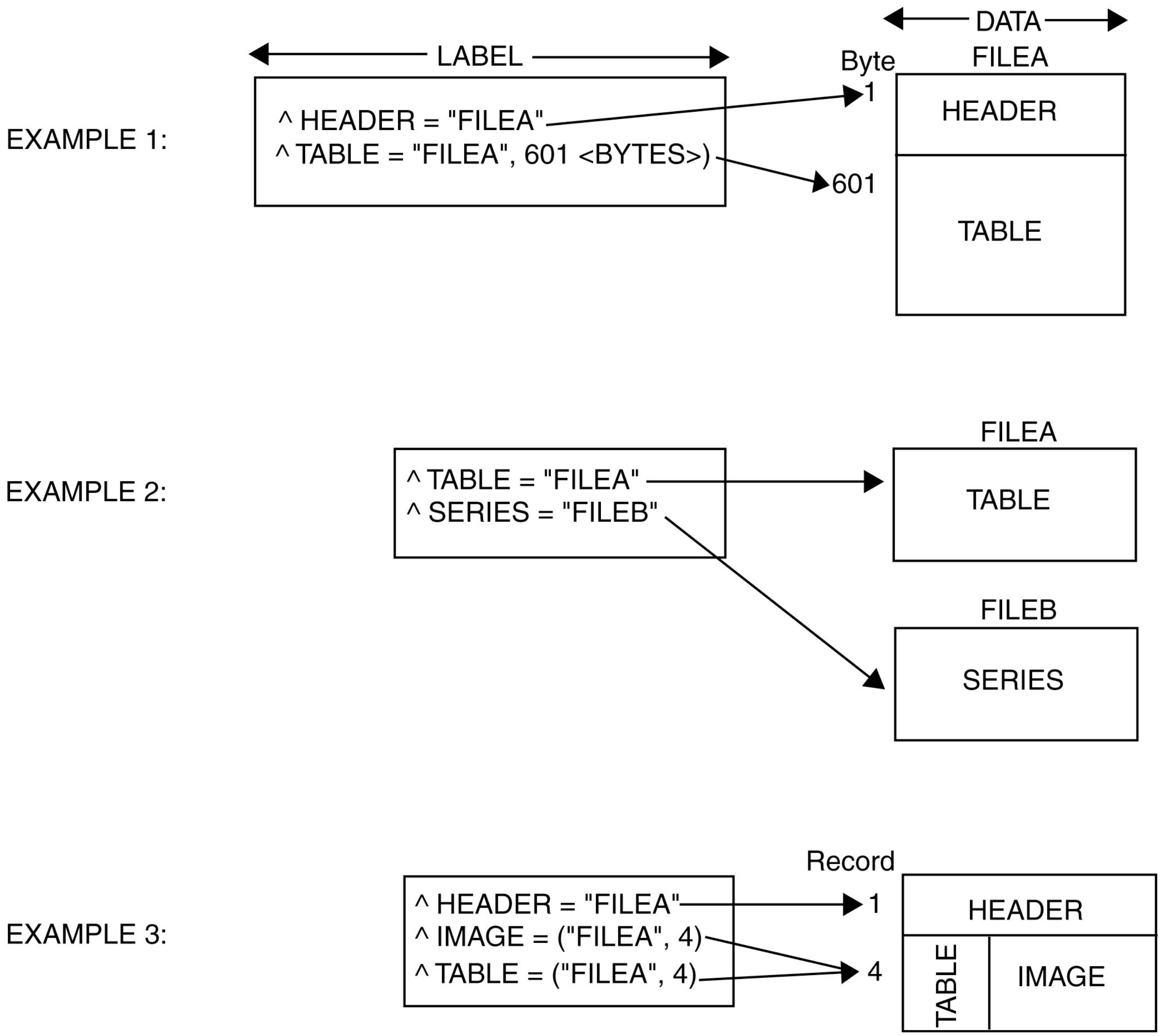


Figure 5.6 Data Object Pointers – Detached & Combined Labels

5.3.4 Data Identification Elements

The data identification elements provide additional information about a data product that can be used to relate the product to other data products from the same data set or data set collection. The minimum set of identification elements required by the PDS standards (see the following subsections) is sufficient to populate a high-level database like, for example, the PDS central catalog. In addition, data preparers will choose additional identification elements from the *Planetary Science Data Dictionary* (PSDD) to support present and future cataloging and search operations.

NOTE: When a data preparer desires a new element for a data product label - one not yet recorded in the PSDD - it can be proposed for addition to the dictionary. Contact a PDS Data Engineer for assistance.

5.3.4.1 Spacecraft Science Data Products

The following data identification elements must be included in product labels for all spacecraft science data products:

DATA_SET_ID
PRODUCT_ID
INSTRUMENT_HOST_NAME
INSTRUMENT_NAME
TARGET_NAME
START_TIME
STOP_TIME
SPACECRAFT_CLOCK_START_COUNT
SPACECRAFT_CLOCK_STOP_COUNT
PRODUCT_CREATION_TIME

5.3.4.2 Earthbased Science Data Products

The following data identification elements must be included in product labels for all Earth-based science data products:

DATA_SET_ID
PRODUCT_ID
INSTRUMENT_HOST_NAME
INSTRUMENT_NAME
TARGET_NAME
START_TIME
STOP_TIME
PRODUCT_CREATION_TIME

5.3.4.3 Ancillary Data Products

The following data identification elements must be included in product labels for all ancillary data products. Ancillary products may be more general in nature, supporting a wide variety of instruments for a particular mission. For example, SPICE data sets, general engineering data sets, and uplink data are considered ancillary data products.

DATA_SET_ID
PRODUCT_ID
PRODUCT_CREATION_TIME

The following identification elements are highly recommended, and should be included in ancillary data products whenever they apply:

INSTRUMENT_HOST_NAME
INSTRUMENT_NAME
TARGET_NAME
START_TIME
STOP_TIME
SPACECRAFT_CLOCK_START_COUNT
SPACECRAFT_CLOCK_STOP_COUNT

5.3.5 Descriptive Data Elements

In addition to the data identification elements required for various types of data, PDS strongly recommends including additional data elements related to specific types of data. These descriptive elements should include any elements needed to interpret or process the data objects or which would be needed to catalog the data product to support potential search criteria at the product level.

Recommendations for descriptive data elements to be included come from the PDS mission interface personnel as well as the data producer's own suggestions. These additional data elements are selected from the *Planetary Science Data Dictionary*.

NOTE: When a data element is needed for a data product label, but is not yet recorded in the PSDD, it may be proposed for addition to the dictionary. Contact a PDS data engineer for assistance in submitting new data elements for inclusion in the PSDD.

Pointers are sometimes used in a PDS label to provide a shorthand method for referencing either a set of descriptive data elements (e.g., ^DESCRIPTION) or a long descriptive text passage relevant to several data product labels.

5.3.6 Data Object Definitions

The PDS requires a separate data object definition within the product label for each object in the product, to describe the structure and associated attributes of each constituent object. Each object definition, whether for a primary or a secondary object, must have a corresponding object pointer as described in Section 5.3.3.

Object definitions are of the form:

```
OBJECT              = aaa        where aaa is the name of the data object
...
END_OBJECT          = aaa
```

The PDS has designed a set of standard data object definitions to be used for labeling products. Among these standard objects are those designed to describe structures commonly used for scientific data storage. Appendix A provides the complete set of PDS object definition requirements, along with examples of product labels.

Pointers are sometimes used in a PDS label to provide a shorthand method for including a standard set of sub-objects referenced in several data product labels. For example, a pointer called "^STRUCTURE" is often used to include a set of COLUMN sub-objects for a TABLE structure used in many labels of the same data set.

5.3.7 End Statement

The END statement ends a PDS label. Where required by an outside agency, the END statement may be followed by one or more SFDU labels.

The PDS does not require SFDU labels on individual products, but they may be required to conform with specific project or other agency requirements. If SFDUs are provided on a data product, they must follow the standards described in the *SFDU Usage* chapter in this document. In some, but not all cases, another SFDU label is required after the PDS END statement to provide “end label” and sometimes “start data” information.

5.4 Syntax for Element Values

The values of keywords must be expressed in a manner appropriate to the type of the keyword. Data types for element values are specified in the element definitions contained in the PSDD. The syntax rules for expressing these values in PDS labels are discussed in detail in Section 12.3 of Chapter 12: *Object Description Language Specification and Usage*. A brief summary is provided here for reference.

Character Strings

Character strings are enclosed in double quotes unless they consist entirely of uppercase letter, number, and/or underscore (_) characters.

Examples

```
NAME        = FILTER                        Correct
NAME        = "FILTER WAVELENGTH"           Correct
NAME        = FILTER_WAVELENGTH             Correct

NAME        = FILTER WAVELENGTH             Incorrect
```

Integers

Integer values must be presented as a string of digits, optionally preceded by a sign. Specifically, no comma or point should be used to group digits. Values that are to be interpreted as integers must not be enclosed in quotation marks of any kind.

Examples

```
ITEMS                  = 12                 Correct
REQUIRED_STORAGE_BYTES = 43364              Correct

ITEMS                  = "12"               Incorrect
REQUIRED_STORAGE_BYTES = 43,364             Incorrect
```

Floating-Point Numbers

Real data values may be expressed as either floating-point numbers with a decimal point or in scientific notation with an exponent. Scientific notation is formatted in the standard manner for program I/O, using the letter “E” as an exponentiation operator. Values that are to be interpreted as real numbers must not be enclosed in quotation marks of any kind.

Examples

```
TELESCOPE_LATITUDE       = 33.476               Correct
TELESCOPE_LATITUDE       = 3.3476E+01           Correct

TELESCOPE_LATITUDE       = "33.476"             Incorrect
TELESCOPE_LATITUDE       = 3.3476 x 10^01       Incorrect
```

Dates and Times

Date and time values must be in the PDS standard date/time format: *YYYY-MM-DD*T*hh:mm:ss.sss*. Date and time values must never be enclosed in quotes of any kind.

Examples

```
START_TIME  = 1990-08-01T23:59:59               Correct

START_TIME  = "1990-08-01T23:59:59"             Incorrect
```

5.5 Locally-defined Data Elements

The PSDD contains a large set of common (global) data elements (keywords) and small sets of locally-defined data elements. The set of common data elements are available for use in any label. Locally-defined data elements may only be used in data product labels.

5.5.1 Justification for Locally-defined Data Elements

There are two justifications for when a locally-defined keyword can be created:

a) the scope of use is limited / local to a small set of data products within a single mission or campaign, or is so specific that only a very few data providers would make use of the locally-defined data element (keyword).

   Examples of data elements in the PSDD having limited scope:

   MAXIMUM_B1950_RING_LONGITUDE [PDS-RINGS]
   The maximum_B1950_ring_longitude element specifies the maximum inertial longitude within a ring area relative to the B1950 prime meridian, rather than to the J2000 prime meridian. The prime meridian is the ascending node of the planet's invariable plane on the Earth's mean equator of B1950. Longitudes are measured in the direction of orbital motion

along the planet's invariable plane to the ring's ascending node, and thence along the ring plane. Note: For areas that cross the prime meridian, the maximum ring longitude will have a value less than the minimum ring longitude.

INSTRUMENT_FORMATTED_DESC [PDS-CN]
The instrument_formatted_desc element contains the formatted instrument descriptions. These descriptions represent the information collected for the PDS Version 1.0 instrument model and were created by extracting instrument information from several tables in the catalog data base.
These descriptions represent an archive since the tables have been eliminated as part of the catalog streamlining task.

DATA_SET_LOCAL_ID [PDS-SBN]
The DATA_SET_LOCAL_ID element provides a short (of order 3 characters) acronym used as the local ID of a data set (Example value: IGLC). It may also appear as the first element of file names from a particular DATA_SET (Example value:IGLCINDX.LBL).

b) the common instance, and any other local instances, currently defined in the PSDD are inadequate in some descriptive capacity:

- the data element definition is too restrictive or inappropriate
- the length of the keyword-value is too short
- different types of units

A possible scenario for the above could be that the Cassini mission wants to use the DATA_QUALITY_ID keyword.

DATA_QUALITY_ID [PSDD] - CHAR(3)
The data_quality_id element provides a numeric key which identifies the quality of data available for a particular time period. The data_quality_id scheme is unique to a given instrument and is described by the associated data_quality_desc element.

But, the Cassini mission wants to re-use the data element in a way that is different from the instance(s) currently defined in the PSDD.

DATA_QUALITY_ID [CASSINI] - CHAR(50)
The data_quality_id element provides a short acronym or identifier of the qualitative state in which the data resided when the data was generated by the instrument team. The data_quality_id is unique to the Cassini mission and is described by the associated data_quality_desc element.

5.5.2 Identification of Locally-defined Data Elements

Locally-defined instances of data elements (keywords) are identified in data product labels as:

<namespace>:<keyword_name>

where <namespace> is the unique namespace to which the keyword is designated.

<keyword_name> is the name of the keyword being included in the data product label.

If there are multiple instances of a keyword, then the specific instance of use is identified as follows:

Example:

```
TARGET_NAME = "EARTH"              (namespace = PSDD)
CASSINI:TARGET_NAME = "EARTH"      (namespace = CASSINI)
VOYAGER:TARGET_NAME = "MARS"       (namespace = VOYAGER)
```

In the above example, the PSDD contains three separate instances of the TARGET_NAME keyword:

a) the common (PSDD) instance which the PDS defined and which the PDS community at large agreed upon.
b) the CASSINI instance which the Cassini project defined.
c) the VOYAGER instance which the Voyager project defined.

5.5.3 Review and Use of Locally-defined Data Elements

The following are recommendations on the review and use of locally defined keywords:

1. The custodian of a namespace is to be a PDS node; or the entity to which the PDS node delegates authority (e.g., mission); or other agencies in a cooperative agreement with NASA and working with the PDS (e.g., ESA).

2. The custodian has initial responsibility for NAMESPACE and all locally defined elements which use the NAMESPACE.

3. The responsibility for NAMESPACE may be transferred if agreeable to the custodian and the receiving party.

4. The responsibility for locally defined elements may be transferred if agreeable to the custodian and the receiving party.

5. Custodians (e.g., missions/campaigns) being phased out are expected to transfer all responsibilities to a continuing party (i.e., there is always a responsible party actively engaged in overseeing the use of the NAMESPACE and locally defined elements which use the NAMESPACE.

6. Control authority (responsible party) has absolute authority over element definitions.

7. A non-originating user who reuses a locally defined keyword must conform to interpretations of the control authority, including retroactive adjustments (i.e., the user of a locally defined keyword is at risk that the control authority may alter one or more of the keyword attributes; such as the definition, without notifying outside users of the change).

8. PDS recommends that non-originating users "clone" elements into a new local dictionary rather than reusing them (e.g., CASSINI:DATA_QUALITY_ID would become MER:DATA_QUALITY_ID if reused by the MER mission). This is because non-originating users are at risk that the keyword may be altered by the control authority and the control authority does not have an obligation to notify anyone of the change.

9. 'Promoting' locally defined keywords to full PSDD standing is not permitted (i.e., locally defined keywords remain locally defined throughout the life of the keyword). A locally defined keyword may be proposed independently for use in the global PSDD by submitting the keyword element definition for the full PSDD approval process.

10. Locally defined keywords should not take on a scope outside of the originating mission/campaign.

(This page intentionally left blank.)

Chapter 6. Data Set / Data Set Collection Contents and Naming

The Data Set / Data Set Collection Contents and Naming standard defines the conventions for maintaining consistency in the contents, organization and naming of archive quality data sets.

Data Sets are defined in terms of Data Products, which were introduced in Chapter 4. A data set is an aggregation of data products with a common origin, history, or application. A data set includes primary (observational) data plus the ancillary data, software, and documentation needed to understand and use the observations. Files in a data set share a unique data set name, share a unique data set identifier, and are described by a single DATA_SET catalog object (or equivalent).

Data Set Collections are defined in terms of data sets. A data set collection is an aggregation of several data sets that are related by observation type, discipline, target, or time which are to be treated as a unit; that is, they are intended to be archived and distributed together. Data sets in a data set collection share a unique data set collection name, share a unique data set collection identifier, and are described by a single DATA_SET_COLLECTION object (or equivalent). One of the primary considerations in creating a data set collection is that the collection as a whole provides more utility than the sum of the utilities of the individual data sets.

Figure 6.1 shows the relationships among Data Products, Data Sets, and a Data Set Collection.

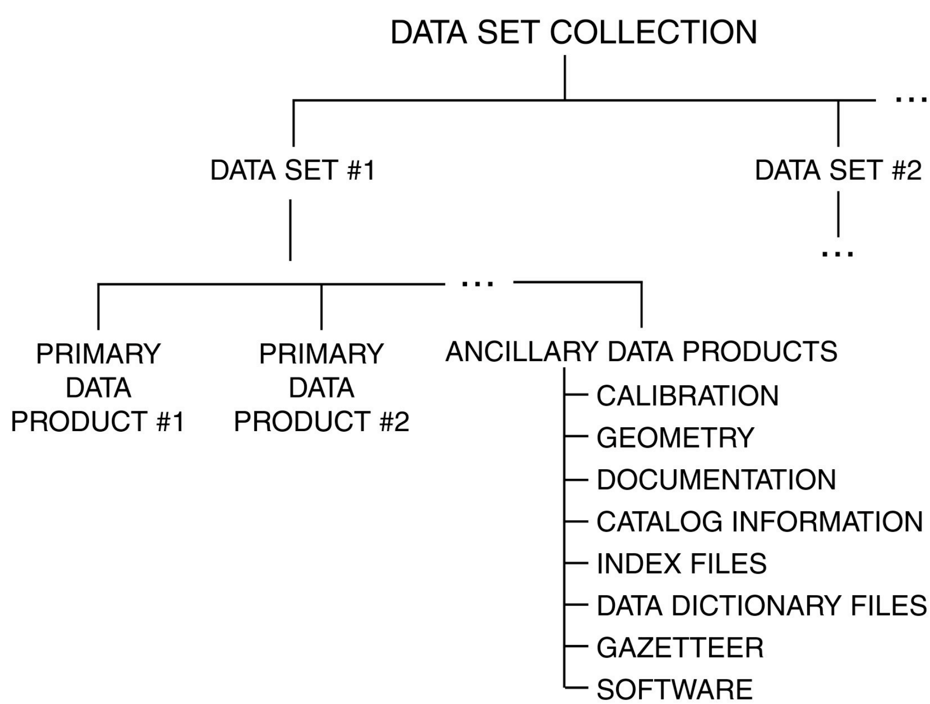


Figure 6.1 Relationships among a Data Set Collection, its Data Sets, and their Data Products.

Note that with respect to Figure 6.1, additional data sets (e.g., Data Set #2) have structure similar to Data Set #1. And, Ancillary Data Products are often organized into directories corresponding to the subject areas shown (see Chapter 19 for a more detailed description of each directory).

Ancillary Data Products may include any or all of the following:

Calibration - Data products used in the conversion of raw measurements to physically meaningful values or data products needed to use the data.

Geometry - Data products needed to describe the observing geometry. Examples include SEDRs and SPICE files.

Documentation - Data products which describe the mission, spacecraft, instrument, and/or data set. These may include references to science papers or the papers themselves.

Catalog Information - Descriptive information about a data set expressed in Object Description Language (ODL) and suitable for loading into a catalog. For more information, see Appendix B.

Index Files - Information that allows a user to locate the data of interest - a table of contents. An example might be a table mapping latitude/longitude ranges to file names.

Data Dictionary Files - An extract of the *Planetary Science Data Dictionary* (PSDD) that is pertinent to the data set and expressed in ODL.

Gazetteer - Information about the named features on a target body associated with the data set.

Software - Software libraries, utilities, and/or application programs to access/process the data products.

6.1 Data Set Naming and Identification

Each PDS data set must have a unique name (DATA_SET_NAME) and a unique identifier (DATA_SET_ID), both formed from up to seven components. The components are listed here; valid assignments for each component are described in Section 6.3:

Instrument host
Target
Instrument
Data processing level number
Data set type (optional)
Description (optional)

Version number

A DATA_SET_NAME must not exceed 60 characters in length. Where the character limitation is not exceeded, the full-length name of each component is used. If the full-length name is too long, an acronym is used to abbreviate components of the name. Where possible, each component of the DATA_SET_NAME should identify and reflect the corresponding (acronym) component used in forming the DATA_SET_ID.

The DATA_SET_ID cannot exceed 40 characters in length. Each component of the DATA_SET_ID is an acronym that identifies and reflects the corresponding (full-name) component used in forming the DATA_SET_NAME. Within the DATA_SET_ID, acronyms are separated by hyphens.

Multiple instrument hosts, instruments, or targets are referenced in a DATA_SET_NAME or DATA_SET_ID by concatenation of the values with a forward slash, "/", which is interpreted as "and." The slash may not be used in any other capacity in a DATA_SET_ID.

6.2 Data Set Collection Naming and Identification

Each PDS data set collection must have a unique name (DATA_SET_COLLECTION_NAME) and a unique identifier (DATA_SET_COLLECTION_ID), both formed from up to six components. A data set collection may contain data sets that cover several targets, be of different processing levels, or have different instrument hosts and instruments. Since the individual data sets will be identified by their own data set names, some of this information need not be repeated at the collection level. Therefore, the DATA_SET_COLLECTION_NAME uses a subset of the DATA_SET_NAME components in addition to a new component, the collection name, which identifies the group of related data sets. The components are listed here; valid assignments for each component are described in Section 6.3:

Collection name
Target
Data processing level number (optional)
Data set type (optional)
Description (optional)
Version number

A DATA_SET_COLLECTION_NAME must not exceed 60 characters in length. Where the character limitation is not exceeded, the full-length name of each component is used. If the full-length name is too long, an acronym should be substituted. Where possible, each component of the DATA_SET_COLLECTION_NAME should identify and reflect the corresponding (acronym) component used in forming the DATA_SET_COLLECTION_ID.

The DATA_SET_COLLECTION_ID must not exceed 40 characters in length. Each component is an acronym that identifies and reflects the corresponding (full-name) component used in forming the DATA_SET_COLLECTION_NAME.

Multiple targets or data processing levels are referenced in the data set collection name or identifier by concatenation of the values with a forward slash (/) which is interpreted as "and."

6.3 Name and ID Components

6.3.1 Restrictions on DATA_SET_ID and DATA_SET_COLLECTION_ID

Within the DATA_SET_ID and DATA_SET_COLLECTION_ID, acronyms are separated by hyphens. The only characters allowed are:

- Uppercase characters, A-Z
- Digits, 0-9
- The hyphen character, "-"
- The forward slash, "/"
- The period character, ".", but only as part of a numeric component (e.g., "V1.0" but not "C.A")

6.3.2 Standard Acronyms, Abbreviations, and Assignments

This section details the standard acronyms and abbreviations required for formulating the DATA_SET_ID and DATA_SET_COLLECTION_ID values. They are also recommended for use, as appropriate, in the formation of other NAME- and ID-class element values. Standard values for data dictionary elements mentioned in the following sections are listed in the PSDD. New values are added to these lists as needed by the PDS data engineers.

1. **Instrument host** name and ID values are selected from the standard value list of the corresponding PSDD entry (INSTRUMENT_HOST_NAME or INSTRUMENT_HOST_ID data element). Note that the acronym EAR has been used for Earth-based data sets without a specific instrument host.

2. **Collection names and IDs** are created as needed by the data preparers in conjunction with the PDS data engineer. Current IDs and their corresponding names include:

GRSFE	Geological Remote Sensing Field Experiment
IHW	International Halley Watch
PREMGN	Pre-Magellan

3. **Target name** values are selected from the standard values listed in the PSDD for the TARGET_NAME element. Target acronyms are selected from the following list:

Target ID	Target Name
A	Asteroid
C	Comet
CAL	Calibration
D	Dust
E	Earth
H	Mercury
J	Jupiter
L	Moon
M	Mars
MET	Meteorite
N	Neptune
P	Pluto
R	Ring
S	Saturn
SA	Satellite
SS	Solar System
U	Uranus
V	Venus
X	Other, (e.g., Checkout)
Y	Sky

NOTE: Satellites or rings are referenced in DATA_SET_NAMEs and DATA_SET_IDs by the concatenation of the satellite or ring identifier with the associated planet identifier; for example:

JR	Jupiter's rings
JSA	Jupiter's satellites

If Jupiter data are also included in the ring and/or satellite data set then only Jupiter ("J") is referenced as the target.

Note that in some cases this component represents the TARGET_TYPE rather than the target name, for example:

A	Asteroid
C	Comet
CAL	Calibration
MET	Meteorite

Valid values for the TARGET_TYPE data element are listed in the PSDD.

4. **Instrument name and ID** values are taken either from the corresponding PSDD element, or from the following list of values designated for certain types of ancillary data:

Names:	INSTRUMENT_NAME data element in the PSDD
IDs:	INSTRUMENT_ID data element in the PSDD
Ancillary Data:	ENG or ENGINEERING for engineering data sets
	SPICE for SPICE data sets
	GCM for Global Circulation Model data
	SEDR for supplemental EDR data
	POS for positional data

5. **Data processing level number** is the National Research Council (NRC) Committee on Data Management and Computation (CODMAC) data processing level number.

   Normally a data set contains data of one processing level. PDS recommends that data of different processing levels be treated as different data sets. However, if it is not possible to separate the data, then a single data set with multiple processing levels will be accepted. Use the following guidelines when specifying the data processing level number component of the data set identifier and name:

   (a) the processing level number of the largest subset of data or
   (b) the highest processing level number if there is no predominant subset.

Level	Type	Data Processing Level Description
1	Raw Data	Telemetry data with data embedded.
2	Edited Data	Corrected for telemetry errors and split or decommutated into a data set for a given instrument. Sometimes called Experimental Data Record. Data are also tagged with time and location of acquisition. Corresponds to NASA Level 0 data.
3	Calibrated Data	Edited data that are still in units produced by instrument, but that have been corrected so that values are expressed in or are proportional to some physical unit such as radiance. No resampling, so edited data can be reconstructed. NASA Level 1A.
4	Resampled Data	Data that have been resampled in the time or space domains in such a way that the original edited data cannot be reconstructed. Could be calibrated in addition to being resampled. NASA Level lB.
5	Derived Data	Derived results, as maps, reports, graphics, etc. NASA Levels 2 through 5.
6	Ancillary Data	Nonscience data needed to generate calibrated or resampled data sets. Consists of instrument gains, offsets, pointing information for scan platforms, etc.
7	Correlative Data	Other science data needed to interpret space-based data sets. May include ground-based data observations such as soil type or ocean buoy measurements of wind drift.
8	User Description	Description of why the data were required, any peculiarities associated with the data sets, and enough documentation to allow secondary user to extract information from the data.
N	N	Not Applicable

6. **Data set type** provides additional identification if, for example, the CODMAC data processing level component is not sufficient to identify the type or level of data. Following is a list of valid IDs and names that may be used for this component.

NOTE: Several of the values in this table are currently unique to a particular mission (e.g., BIDR and MIDR were used on Magellan). These values may be used on other missions, if deemed appropriate.

ID	Name
ADR	Analyzed Data Record
BIDR	Basic Image Data Record
CDR	Composite Data Record
CK	SPICE CK (Pointing Kernel)
DDR	Derived Data Record (possibly multiple instruments)
DIDR	Digitalized Image Data Record
DLC	Detailed Level Catalog
EDC	Existing Data Catalog
EDR	Experiment Data Record
EK	SPICE EK (Event Kernel)
FK	SPICE FK (Frames Kernel)
GDR	Global Data Record
IDR	Intermediate Data Record
IK	SPICE IK (Instrument Kernel)
LSK	SPICE LSK (Leap Second Kernel)
MDR	Master Data Record
MIDR	Mosaicked Image Data Record
ODR	Original Data Record
PCK	SPICE PCK (Planetary Constants Kernel)
PGDR	Photograph Data Record
RDR	Reduced Data Record
REFDR	Reformatted Data Record
SDR	System Data Record
SEDR	Supplementary Experiment Data Record
SPK	SPICE SPK (Ephemeris Kernel)
SUMM	Summary (data) (to be used in the browse function)
SAMP	Sample data from a data set (not subsampled data)

7. **Description** is optional, but allows the data provider to describe the data set better – for example, to identify a specific comet or asteroid. Following is a list of example values (both IDs and names) that can be used for this component.

ID	Name
ALT/RAD	Altimetry and Radiometry
BR	Browse
CLOUD	Cloud
ELE	Electron
ETA-AQUAR	Eta-Aquarid Meteors
FULL-RES	Full Resolution
GIACOBIN-ZIN	Comet P/Giacobini-Zinner
HALLEY	Comet P/Halley
ION	Ion
LOS	Line of Sight Gravity
MOM	Moment
PAR	Parameter
SA	Spectrum Analyzer
SA-4.0SEC	Spectrum Analyzer 4.0 second
SA-48.0SEC	Spectrum Analyzer 48.0 second

8. **Version number** is determined as follows:

   (a) If there is not a previous version of the PDS data set/data set collection, then use Version 1.0.

   (b) If a previous version exists, then PDS recommends the following:

      i. If the data sets/data set collections contain the same set of data, but use a different medium (e.g., CD-ROM), then no new version number is required (i.e., no new data set identifier). The inventory system will handle the different media for the same data set.

      ii. If the data sets/data set collections contain the same set of data, but have minor corrections or improvements such as a change in descriptive labeling, then the version number is incremented by a tenth. For example, V1.0 becomes V1.1.

      iii. If a data set/data set collection has been reprocessed, using, for example, a new processing algorithm or different calibration data, then the version number is incremented by one (V1.0 would become V2.0). Also, if one data set/data set collection contains a subset, is a proper subset, or is a superset of another, then the version number is incremented by one.

6.4 Examples

For a data set containing the first version of Mars Cloud Data derived from the Mariner 9, Viking Orbiter 1, and Viking Orbiter 2 imaging subsystems, the data set name and identifier would be:

```
DATA_SET_NAME = "MR9/V01/V02 MARS ISS/VIS 5 CLOUD V1.0"
DATA_SET_ID   = "MR9/V01/V02-M-ISS/VIS-5-CLOUD-V1.0"
```

In this example the optional data set type is not used. The other components are:

- Instrument hosts are Mariner 9, Viking Orbiter 1 and Viking Orbiter 2
- Target is Mars
- Instruments are the Imaging Science Subsystem and Visual Imaging Subsystem
- Data Processing Level number is 5
- Description is CLOUD
- Version number is V1.0

Note that the individual components in the DATA_SET_ID closely match the corresponding components used in the DATA_SET_NAME.

The Pre-Magellan Data Set Collection contains radar and gravity data similar to the kinds of data that Magellan collected and was used for pre-Magellan analyses of Venus and for comparisons to actual Magellan data. In conversation the data set might be described as Pre-Magellan Earth, Moon, Mercury, Mars, and Venus Resampled and Derived Radar and Gravity Data Version 1.0. The data set collection name and ID were:
:

```
DATA_SET_COLLECTION_NAME = "PRE-MAGELLAN E/L/H/M/V 4/5 RADAR/GRAVITY
                            DATA V1.0"

DATA_SET_COLLECTION_ID   = "PREMGN-E/L/H/M/V-4/5-RAD/GRAV-V1.0"
```

(This page intentionally left blank.)

Chapter 7. Date/Time Format

PDS has adopted a subset of the International Standards Organization Standard (ISO/DIS) 8601 standard entitled "Data Element and Interchange Formats - Representations of Dates and Times", and applies the standard across all disciplines in order to give the system generality. See also Dates and Times in *Object Description Language* (Chapter 12, Section 12.3.2) of this document.

It is important to note that the ISO/DIS 8601 standard covers only ASCII representations of dates and times.

ODL Date/Time Information
Chapter 12, *Object Description Language (ODL) Specification and Usage*, Section 12.3.2, **Dates and Times**, of this document provides additional information on the use of ODL in date/time formation, representation, and implementation.

7.1 Date/Times

In the PDS there are two recognized date/time formats:

CCYY-MM-DDTHH:MM:SS.sss (preferred format)
CCYY-DDDTHH:MM:SS.sss

Each format represents a concatenation of the conventional date and time expressions with the two parts separated by the letter T:

CC	-	century (00-99)
YY	-	year (00-99)
MM	-	month (01-12)
DD	-	day of month (01-31)
DDD	-	day of year (001-366)
T	-	date/time separator
HH	-	hour (00-23)
MM	-	minute (00-59)
SS	-	second (00-59)
sss	-	fractions of second (000-999)

Note: See Section 7.4 "Midnight and Leap Seconds" for special cases involving the indication of midnight and leap seconds.

The preferred date/time format is: CCYY-MM-DDTHH:MM:SS.sss.

Date/Time Precision

The above date/time formats may be truncated on the right to match the precision of the date/time value in any of the following forms:

1998
1998-12
1998-12-01
1998-12-01T23
1998-12-01T23:59
1998-12-01T23:59:58
1998-12-01T23:59:58.1
1998-12-01T23:59:58.12

7.2 Dates

Dates should be expressed in the conventional ISO/DIS 8601 format. On those rare occasions when dates cannot be expressed in the conventional format, a native format may be used.

7.2.1 Conventional Dates

Conventional dates are represented in ISO/DIS 8601 format as either year (including century), month, day-of-month (CCYY-MM-DD), or as year, day-of-year (CCYY-DDD). The hyphen character ('-') is used as the field separator in this format. The year, month, day-of month format is the preferred format for use in PDS labels and catalog files and is referred to as *PDS standard date format*, but either format is acceptable.

7.2.2 Native Dates

Dates in any format other than the ISO/DIS 8601 format described above are considered to be in a format native to the specific data set, thus "native dates". Native date formats are specified by the data preparer in conjunction with the PDS data engineer. Mission-elapsed days and time-to-encounter are both examples of native dates.

7.3 Times

The PDS allows times to be expressed in conventional and native (alternate) formats.

7.3.1 Conventional Times

Conventional times are represented as hours, minutes and seconds according to the ISO/DIS 8601 time format standard: HH:MM:SS[.sss]. Note that the hours, minutes, and integral seconds fields must contain two digits. The colon (':') is used as a field separator. Fractional seconds consisting of a decimal point (the European-style comma may not be used) and up to three digits (thousandths of a second) may be included if appropriate.

Coordinated Universal Time (UTC) is the PDS time standard and must be formatted in the previously described ISO/DIS 8601 standard format. The letter "Z", indicating the civil time zone at Greenwich (i.e., GMT), should never be appended to a UTC time. The relationship between UTC and GMT has varied historically and with observer context. Note that in PDS data sets created under earlier versions of the Standards, an appended “Z” is taken as indicating UTC.

The START_TIME and STOP_TIME data elements required in data product labels and catalog files are in UTC. For data collected by spacecraft-mounted instruments, the date/ time must be a time that corresponds to “spacecraft event time”. For data collected by instruments not located on a spacecraft, this time shall be an earth-based event time.

Adoption of UTC (rather than spacecraft-clock-count, for example) as the standard facilitates comparison of data from a particular spacecraft or ground-based facility with data from other sources.

7.3.2 Native Times

Times in any format other than the ISO/DIS 8601 format described above are considered to be in a format native to the data set, and thus “native times”. The NATIVE_START_TIME and NATIVE_STOP_TIME elements hold the native time equivalents of the UTC values in START_TIME and STOP_TIME, respectively.

There is one native time of particular interest, however, which has specific keywords associated with it. The spacecraft clock reading (that is, the “count”) often provides the essential timing information for a space-based observation. Therefore, the elements SPACECRAFT_CLOCK_START_COUNT and SPACECRAFT_CLOCK_STOP_COUNT are required in labels describing space-based data. This value is formatted as a string to preserve precision.

Note that in rare cases in which there is more than one native time relevant to an observation, the data preparer should consult a PDS data engineer for assistance in selecting the appropriate PDS elements.

Examples of quantities that may be expressed in native time formats include:

1. Spacecraft Clock Count (sclk)
2. Ephemeris Time
3. Relative Time
4. Local Time

7.4 Midnight and Leap Seconds

The ISO/DIS 8601 standard for representation of midnight and leap seconds are also used in PDS time fields.

7.4.1 Midnight

Midnight may be indicated in one of two ways: as “00:00:00” or “24:00:00”. The usual precision modifications apply as well – i.e. “24:00” is also recognized as midnight.

The “00:00:00” notation is used to indicate midnight at the beginning of a date. “24:00:00” is used to indicate midnight at the end of a date. So, for example, the following two date/time strings refer to precisely the same moment:

2007-04-07T24:00:00 = 2007-04-08T00:00:00

When the hours field has the value “24”, any and all subsequent time fields *must* be zero.

7.4.2 Leap Seconds

Leap seconds may be positive or negative, but in either case are always applied at the end of the day in question. A positive leap second is indicated with a time value of “23:59:60”. A negative leap second is indicated by the omission of the time “23:59:59”. That is, on the day of a negative leap second, the sequence leading through midnight is:

23:59:57
23:59:58
00:00:00
00:00:01

And on the day of a positive leap second, the sequence through midnight is:

23:59:58
23:59:59
23:59:60
00:00:00
00:00:01

Note that the only time when the seconds value of a time string may contain the value “60” is when this represents a positive leap second.

Chapter 8. Directory Types and Naming

The Directory Naming standard defines the conventions for naming directories on a data volume. This chapter lists the standard directories established by PDS, plus the rules for forming subdirectory names and abbreviations.

8.1 Standard Directory Names

When any of the following directories are included on an archive product, the following standard directory naming conventions are used.

Directory	***Contents***
CATALOG	PDS catalog files
DOCUMENT	Documentation, supplementary and ancillary information to assist in understanding and using the data products
EXTRAS	"Value added" elements included by the data preparer, but outside the scope of the PDS archive requirements
GAZETTER	Tables of information about the geological features of a target
INDEX	Indices to assist in locating data of interest
LABEL	"Include" files which describe specific aspects of the data format and organization
SOFTWARE	Utilities, application programs, or subprograms used to access or process the data

The following standard directory names are recommended for use on archive volumes. Note that these directory names are reserved for the uses described below. That is, if they appear on an archive volume, they must contain the indicated information:

CALIB	Calibration files used in the original processing of the data, or needed to use the data
GEOMETRY	Files describing the observational geometry (e.g., SEDRs, SPICE kernels)
BROWSE	Reduced resolution versions of data products
DATA	Contains one or more subdirectories of data products. The DATA subdirectory is used to unclutter the root directory of a volume by providing a single entry point to multiple data subdirectories.

Note that some data sets may not contain all the components above and, as a result, do not need all of the directories listed. For example, many image data sets do not include geometry files and so do not need a GEOMETRY directory. See the *Volume Organization and Naming* chapter of this document for a list of required and optional subdirectories on any specific volume.

8.2 Formation of Directory Names

1. A directory name must consist of only uppercase alphanumeric characters and the underscore character (i.e., A-Z, 0-9, or "_"). No lowercase letters (i.e., a-z) or special characters (e.g., "#", "&", "*") are allowed.

2. Directory names must comply with the ISO 9660 Level 2 standard and not exceed 31 characters in length. Users are encouraged to keep directory names as brief as practical in the interests of providing succinct file paths and easy to read directory listings. The total length of the directory path and file name must not exceed 255 characters.

3. The first letter of a directory name must be an alphabetic character, unless the directory name represents a year (e.g., 1984).

4. If numeric characters are used as part of the name (e.g., DIR1, DIR2, DIR3) the numeric part should be padded with leading zeros up to the maximum size of the numeric (DIR0001, DIR0002, DIR3267).

5. Directories which contain a range of similarly named files must be assigned directory names using the portion of the filename which encompasses all the files in the directory, with "X's" used to indicate the range of values of actual filenames in the directory.

   For example, the PDS Uranus Imaging CD-ROM disk contains image files that have filenames that correspond to SPACECRAFT_CLOCK_START_COUNT values. The directory that contains the image files ranging from C2674702.IMG through C2674959.IMG has the directory name C2674XXX.

6. Directory names must use full length terms whenever possible (e.g., SATURN, MAGELLAN, CRUISE, NORTH, DATA, SOFTWARE). Otherwise, directory names must be constructed from abbreviations of full-length names using the underscore character to separate abbreviated terms, if possible. The meaning of the directory name should be clear from the abbreviation and from the directory structure.

For example, the following directory structure can be found on the Voyager 2 Images of Uranus CD-ROM Volume 1:

```
ROOT ───┬─── ARIEL
        ├─── DOCUMENT
        ├─── INDEX
        ├─── OBERON
        ├─── TITANIA
        ├─── UMBRIEL
        ├─── UNKNOWN
        ├─── URANUS ─┬─── C2674XXX
        │            ├─── C2675XXX
        │            │    …
        │            └─── C2687XXX
        ├─── U_RINGS ┬─── C2674XXX
        │            ├─── …
```

In this case, it is clear from the context that the directory U_RINGS is the abbreviated form of URANUS_RINGS.

7. High level directories that deal with data sets covering a range of planetary science disciplines or targets shall adhere to the following hierarchy:

A Planetary science directory:	PLANET
Planetary body subdirectories:	MERCURY, MOON, MARS, VENUS, COMET
Discipline subdirectories:	ATMOS, IONOSPHE, MAGNETOS, RING, SURFACE, and SATELLIT (Use satellite name if numerous files exist)

8. The recommended SOFTWARE subdirectory naming convention is described in the *Volume Organization and Naming* chapter of this document. Either a platform-based model or an application-based model can be used in defining software subdirectories. In a platform-based model, the hardware platform, operating system and environment must be explicitly stated. If there is more than one operating system/environment supported they are addressed as subdirectories under the hardware directories. When there is only one, the subdirectory may be promoted to the hardware directory.

   For example, if software for the PC for both DOS and Windows were present on the volume, the directories SOFTWARE/PC/DOS and SOFTWARE/PC/WIN would exist. If only DOS software were present, the directory would be SOFTWARE/PCDOS.

8.3 Path Formation Standard

The PDS standard for path names is based on Level 2 of the ISO 9660 international standard. A pathname may consist of up to eight directory levels. Each directory name is limited to 31 characters; the forward-slash character ("/") is used as the separator in path names. The total length of the directory path and file name must not exceed 255 characters. Path names typically appear on PDS volumes as data in index tables for locating specific files on an archive volume. They may also appear as values in a limited number of keywords (e.g., FILE_SPECIFICATION_NAME, PATH_NAME, and LOGICAL_VOLUME_PATH_NAME).

The following are examples of valid values for the keywords listed above:

TG15NXXX/TG15N1XX/TG15N12X/	identifies the location of the directory TG15N12X at the third level below the top level of an archive volume.
DOCUMENT/	identifies a DOCUMENT directory within the root directory.

Note: The leading slash is omitted because these are relative paths. The trailing slash is included so that concatenation of PATH_NAME and FILE_NAME will yield the full file specification. See the *File Specification and Naming* chapter of this document for more information.

Previous PDS standards allowed the use of the DEC VMS syntax for path names. While PDS support for this format continues to exist, it is recommended that all future volumes use the UNIX syntax instead.

8.4 Tape Volumes

When magnetic tape is the archive medium, a disk directory structure cannot be used because the medium does not support multi-level directories. In this case, files must be stored sequentially.

A directory structure for the volume must be designed in any case, so that when the data are transferred to a medium that supports hierarchical file management they can be placed into an appropriate directory structure. A DIRECTORY object must be included with each tape volume within the VOLUME object. This object is then used to describe how the sequential files should be loaded into a hierarchical structure.

8.5 Exceptions to These Standards

In certain cases, the archive medium used to store the data, the hardware used to produce the data set, or the software operating on the data may impose restrictions on directory names and organization. In these cases, consult a PDS data engineer for guidance in designing the archive volume structure.

Chapter 9. Documents

Supplementary or ancillary reference materials are usually included with archive products to improve their short- and long-term utility. These documents augment the internal documentation of the product labels and provide further assistance in understanding the data products and accompanying materials. Typical archive documents include:

- Flight project documents
- Instrument papers
- Science articles
- Volume information
- Software Interface Specifications (SISs)
- Software user manuals

The PDS criteria for inclusion of a document in the archive are:

1. Would this information be helpful to a data user?
2. Is the material necessary?
3. Is the documentation complete?

In general, the PDS seeks to err on the side of completeness.

Each document to be archived must be prepared and saved in a PDS-compliant format, including a PDS label. Documents are delivered in the DOCUMENT directory of an archive volume (see the *Volume Organization and Naming* chapter of this document).

A flat, human-readable ASCII text version of each document must be included on the volume, although additional versions may be included in other supported formats at the option of the data producer. "Flat ASCII text" means the file may contain only the standard, 7-bit printable ASCII character set, plus the blank character and the carriage-return and linefeed characters as record delimiters. A file is "human-readable" if it is not encoded and if any special markup tags which may be included do not significantly interfere with an average user's ability to read the file. So, for example, simple HTML files and TeX/LaTeX files with relatively little markup embedded in the text are generally considered human-readable and may, therefore, be used to satisfy the above ASCII text version requirement.

Note that the PDS takes the requirement for complete documentation very seriously. Documents that are essential to the understanding of an archive are considered as important as the data files themselves. Furthermore, including a document in a PDS archive constitutes publication (or re-publication) of that document. Consequently, documents prepared for inclusion in an archive are expected to meet not only the PDS label and format requirements, but also the structural, grammatical and lexical requirements of a refereed journal submission. Documents submitted for archiving which contain spelling errors, poor grammar or illogical organization will be rejected and may ultimately lead to the rejection of the submitted data for lack of adequate documentation.

9.1 PDS Objects for Documents

PDS labels of documentation files use either the TEXT or DOCUMENT object, as appropriate. The DOCUMENT object is usually used with documentation files found in the DOCUMENT directory of an archive volume. Files described by a DOCUMENT object may be in any of the formats described in Section 9.2.

The TEXT object may only be used with ASCII text files containing no markup. TEXT objects are most often used for small text files occurring anywhere in the archive volume (for example, the AAREADME.TXT file in the root directory or the DOCINFO.TXT file in the DOCUMENT directory).

9.1.1 TEXT Objects

TEXT objects are preferred for stand-alone documents with a narrow focus. For example, the AAREADME.TXT or DOCINFO.TXT files on the archive volume are usually labeled using a TEXT object. Files described by a TEXT object must:

a) Be plain, flat ASCII files without markup tags (i.e., no HTML or TeX files), encoded graphics (as in PostScript files), or programmatic structures (i.e., no source code files or scripting commands); and

b) Have a file extension of ".TXT"

9.1.2 DOCUMENT Objects

DOCUMENT objects are preferred when several versions of the same file are provided or when there are several component files constituting a single version of the document - for example, when graphics are included in separate files from the text. Any file labeled using a DOCUMENT object must:

a) Be in one of the PDS-approved formats listed below; and

b) Use the appropriate object characteristics (listed below) for the DOCUMENT object parameters and the file extension.

DOCUMENT labels are most often combined detached labels, since attaching them to most of the formats listed below would make the combined file unusable in its customary environment (Microsoft *Word*, for example, cannot recognize ".DOC" files with attached PDS labels).

Format	Object	Interchange Format	Document Format	File Extension
Plain ASCII Text	ASCII_DOCUMENT	ASCII	TEXT	.ASC
HTML	HTML_DOCUMENT	ASCII	HTML	.HTM or .HTML*
TeX	TEX_DOCUMENT	ASCII	TEX	.TEX
LaTeX	LATEX_DOCUMENT	ASCII	LATEX	.TEX
Adobe PDF	PDF_DOCUMENT	BINARY	ADOBE PDF	.PDF
MS Word	WORD_DOCUMENT	BINARY	MICROSOFT WORD	.DOC
Rich Text	RTF_DOCUMENT	BINARY	RICH TEXT	.RTF
GIF	GIF_DOCUMENT	BINARY	GIF	.GIF
JPG	JPG_DOCUMENT	BINARY	JPG	.JPG
Encapsulated Postscript	EPS_DOCUMENT	BINARY	ENCAPSULATED POSTSCRIPT	.EPS
PNG	PNG_DOCUMENT	BINARY	PNG	.PNG
Postscript	PS_DOCUMENT	BINARY	POSTSCRIPT	.PS
Tagged Image File Format	TIFF_DOCUMENT	BINARY	TIFF	.TIF or .TIFF*

* See chapter *File Specification and Naming* regarding extensions with more than three characters.

Example: “MYDOC” is a documentation file to be included in the DOCUMENT directory of an archive volume. Two versions will be supplied: a flat ASCII version with the graphics in separate TIFF files; and a Microsoft Word version with in-line graphics in a single file. In the PDS label, “MYDOC” will be described using a DOCUMENT object for each different file format provided. The files included in the directory will be:

1. MYDOC.ASC required ASCII version
2. MYDOC.DOC optional Microsoft Word version to retain all graphics
3. MYDOC001.TIF optional scanned TIFF version of selected pages
4. MYDOC002.TIF optional scanned TIFF version of other selected pages
5. MYDOC.LBL PDS label defining DOCUMENT object(s) for these files

Optional versions of the document should have the same file name as the required ASCII version but with different extensions. Optional versions should be defined as additional DOCUMENT objects in the single PDS label; the name of the required ASCII file should be indicated in the text of the DESCRIPTION keyword.

9.2 Document Format Details

9.2.1 Flat ASCII Text

Line Length and Delimiters - PDS recommends plain text files have line length restricted to 78 characters or fewer, to accommodate printing and display on standard devices. Each line must be terminated by the two-character carriage-return/linefeed sequence (ASCII decimal character codes 13 and 10, respectively).

Page Length and Breaks - Block paragraph style is preferred, with paragraphs being separated by at least one blank line. The form feed character (ASCII decimal code 12) may be used to indicate page breaks, in which case pages should contain no more than 60 lines of text. A formfeed character should be inserted immediately after the END statement line of an attached PDS label in these files.

9.2.2 ASCII Text Containing Markup Language

Line Length and Delimiters - The 78-character line length recommendation is dropped for these files. Notwithstanding, the lines must be delimited by the carriage return/linefeed character combination.

Page Length and Breaks - Page breaks are controlled by the markup in these files. Consequently, there are no specific page length recommendations.

Note: ASCII files containing extensive markup may not pass the "human-readable" test. Also, some automatic converters producing, for example, HTML files that might be expected to be human-readable in fact add so many additional marks and notations that those files also fail the "human-readable" test. Consult a PDS data engineer for help in determining whether a particular file can be considered "human-readable" for archive purposes.

9.2.2.1 Hyper-Text Markup Language (HTML) Files

PDS archive products must adhere to Version 3.2 of the HTML language, a standard generalized markup language (SGML) conforming to the ISO 8879 standard. All files are subject to validation against the HTML 3.2 SGML Declaration and the HTML Document Type Definition.

Note: Constructs not defined in the HTML 3.2 standard (e.g., FRAME, STYLE, SCRIPT, and FONT FACE tags) are not allowed in PDS documentation files.

9.2.2.2 Location of Files

PDS strongly recommends that targets of all HTML links be present on the archive volume. In cases where external links are provided, the link should lead to supplementary information that is not essential to understanding or use of the archival data.

PDS recommends that all files comprising an HTML document or series of documents be located in a single directory. However, locating ancillary files (e.g., images, common files) in subdirectories may be required under certain circumstances (e.g., to avoid conflicts in file names or to minimize replication of common files).

9.2.2.3 Discouraged HTML 3.2 Capabilities

Although the APPLET tag is advertised to be supported by all Java enabled browsers, not all applets execute on all browsers on all platforms. Further, some browsers require that the user explicitly enable use of Java applets before the applet will execute. Consequently, applets are permitted in PDS document files only when the information they convey is not essential to understanding or use of the archival data.

Use of the TAB character is permitted but strongly discouraged because of variations in implementation among browsers and resulting misalignments within documents.

Use of animated GIF image files is discouraged.

9.2.3 Non-ASCII Formats

Wherever possible the specific encoding and version level information should be included in the label for all non-ASCII documents. The ENCODING_TYPE keyword is used to indicate the base encoding type (e.g., PostScript, GIF, etc.), while the specific version information should be included in the text of the DESCRIPTION keyword. See the PSDD for a list of standard encoding types. Additional types may be added at the discretion of the PDS data engineer.

9.2.4 Validation

Documentation files prepared to accompany a data set or data set collection must be validated. Validation consists of checking to ensure that the files can be copied or transmitted electronically, and can be read or printed by their target text-processing program. Documentation files should be spell-checked prior to being submitted to PDS for validation.

9.3 Examples

9.3.1 Simple Example of Attached label (Plain ASCII Text)

The following label could be attached to a plain ASCII text file describing the content and format of Mars Pathfinder Imager Experiment Data Records.

```
PDS_VERSION_ID    = PDS3
RECORD_TYPE       = STREAM
OBJECT            = TEXT
NOTE              = "Mars Pathfinder Imager Experiment Data Record SIS"
PUBLICATION_DATE  = 1998-06-30
END_OBJECT        = TEXT
END
```

9.3.2 Complex Example of Detached Label (Two Document Versions)

If the data producer chose to provide the same document in both plain ASCII text and as a Microsoft Word document, the detached label would have the name EDRSIS.LBL and would be as follows:

```
PDS_VERSION_ID          = PDS3
RECORD_TYPE             = UNDEFINED
```

```
^ASCII_DOCUMENT          = "EDRSIS.ASC"
^WORD_DOCUMENT           = "EDRSIS.DOC"

OBJECT                   = ASCII_DOCUMENT
 DOCUMENT_NAME           = "Mars Pathfinder Imager Experiment Data Record"
 PUBLICATION_DATE        = 1998-06-30
 DOCUMENT_TOPIC_TYPE     = "DATA PRODUCT SIS"
 INTERCHANGE_FORMAT      = ASCII
 DOCUMENT_FORMAT         = TEXT
 DESCRIPTION             = "This document contains a textual description of
                            the VICAR and PDS formatted Mars Pathfinder IMP
                            Experiment Data Records. This is the ASCII text
                            version of the document required by PDS."
END_OBJECT               = ASCII_DOCUMENT

OBJECT                   = WORD_DOCUMENT
  DOCUMENT_NAME          = "Mars Pathfinder Imager Experiment Data Record"
  PUBLICATION_DATE       = 1998-06-30
  DOCUMENT_TOPIC_TYPE    = "DATA PRODUCT SIS"
  INTERCHANGE_FORMAT     = BINARY
  DOCUMENT_FORMAT        = "MICROSOFT WORD"
  DESCRIPTION            = "This document contains a textual description of
                            the VICAR and PDS formatted Mars Pathfinder IMP
                            Experiment Data Records. The document was
                            created using Microsoft Word 6.0.1 for the
                            Macintosh."
END_OBJECT               = WORD_DOCUMENT
END
```

9.3.3 Complex Example of Detached Label (Documents Plus Graphics)

The following label (EDRSIS.LBL) illustrates the use of an HTML document as the required ASCII document. The same document is also included as a PDF file, and four PNG images are included separately.

```
PDS_VERSION_ID           = PDS3
RECORD_TYPE              = UNDEFINED
^HTML_DOCUMENT           = "EDRSIS.HTM"
^PDF_DOCUMENT            = "EDRSIS.PDF"
^PNG_DOCUMENT            = ("FIG1.PNG","FIG2.PNG","TAB1.PNG","TAB2.PNG")

OBJECT                   = HTML_DOCUMENT
  DOCUMENT_NAME          = "Mars Pathfinder Imager Experiment Data
                            Record"
  PUBLICATION_DATE       = 1998-06-30
  DOCUMENT_TOPIC_TYPE    = "DATA PRODUCT SIS"
  INTERCHANGE_FORMAT     = ASCII
  DOCUMENT_FORMAT        = HTML
  DESCRIPTION            = "This document contains a description
                            of the VICAR and PDS formatted Mars
                            Pathfinder IMP Experiment Data Records.  This
                            is an HTML version of the document."
END_OBJECT               = HTML_DOCUMENT

OBJECT                   = PDF_DOCUMENT
  DOCUMENT_NAME          = "Mars Pathfinder Imager Experiment Data
```

```
                              Record"
  PUBLICATION_DATE       = 1998-06-30
  DOCUMENT_TOPIC_TYPE    = "DATA PRODUCT SIS"
  ENCODING_TYPE          = "PDS-ADOBE-1.1"
  INTERCHANGE_FORMAT     = BINARY
  DOCUMENT_FORMAT        = "ADOBE PDF"
  DESCRIPTION            = "This document contains a description
                            of the VICAR and PDS formatted Mars
                            Pathfinder IMP Experiment Data Records. This
                            is a PDF version of the document."
END_OBJECT               = PDF_DOCUMENT

OBJECT                   = PNG_DOCUMENT
  DOCUMENT_NAME          = "Mars Pathfinder Imager Experiment Data
                            Record"
  PUBLICATION_DATE       = 1998-06-30
  DOCUMENT_TOPIC_TYPE    = "DATA PRODUCT SIS"
  FILES                  = 4
  ENCODING_TYPE          = "PNG1.0"
  INTERCHANGE_FORMAT     = BINARY
  DOCUMENT_FORMAT        = PNG
  DESCRIPTION            =  "This document is a PNG representation of two
                            figures and two tables from the Mars
                            Pathfinder IMP Experiment Data Record SIS."
END_OBJECT               = PNG_DOCUMENT
END
```

(This page intentionally left blank.)

Chapter 10. File Specification and Naming

The *File Specification and Naming* standard defines the PDS conventions for forming file specifications and names. This chapter is based on levels 1 and 2 of the international standard ISO 9660, “Information Processing - Volume and File Structure of CD-ROM for Information Interchange.”

ISO 9660 Level 1 versus ISO 9660 Level 2

PDS recommends that archive products delivered on physical media adhere to the ISO 9660 Level 1 specification. Specifically, CD-ROM volumes that are expected to be widely distributed should use file identifiers consisting of a maximum of eight characters in the base name and three characters in the extension (i.e., “8.3” file names), as described in Section 10.1.1. When there are compelling reasons to relax the 8.3 file name standard, the ISO 9660 Level 2 specification with respect to file names may be used, subject to the restrictions listed in Section 10.1.2.

Electronic Transfer and Storage of Archives

The ISO 9660 Level 1 and Level 2 standards are specifically for CD and DVD media as described in Section 11. Data providers may negotiate with their PDS Node to deliver archives electronically rather than on CD-ROM. As electronic delivery of archives is becoming a more common practice, PDS is pursuing a more comprehensive standard. Pending the results of this investigation, PDS requires that electronically delivered data meet the restrictions listed in Section 10.1.3.

10.1 File Specification Standards

A file specification consists of the following elements:

1. A complete directory path name (as discussed in the *Directory Types and Naming* chapter of this document)
2. A file name (including extension)

The PDS has adopted the UNIX/POSIX forward slash character (/) as the directory separator for use in path names. Directory path name formation is discussed further in the *Directory Types and Naming* chapter of this document.

The following is an example of a simple file specification. The file specification identifies the location of the file relative to the root of a volume, including the directory path name.

File Name: TG15N122.IMG

File Specification: TG15NXXX/TG15N1XX/TG15N12X/TG15N122.IMG

Do not use path or file names that correspond to operating system specific names, such as:

AUX COM1 CON LPT1 NUL PRN

10.1.1 ISO 9660 Level 1 Specification

A file name consists of a base name and an extension, separated by a full stop character (“.”). Under ISO 9660 Level 1, the length of the base name may not exceed eight characters and the extension may not exceed three characters. In addition, a version number consisting of a semicolon and an integer must follow the file identifier. The base name and extension may only contain characters from the following set: the upper case alphanumeric characters (A- Z, 0-9) and the underscore (“_”). Collectively, these requirements are often referred to as the “8.3” (“8 dot 3”) file naming convention. These limitations exist primarily to accommodate older computer systems that cannot handle longer file names.

Preferred format:	BASENAME (1..8 characters) "." EXTENSION (3 characters)
Allowable format:	BASENAME (1..8 characters) "." EXTENSION (1..3 characters)
Actual format on archive medium:	BASENAME (1..8 characters) "." EXTENSION (1..3 characters) ";1"

10.1.2 ISO 9660 Level 2 Specification

The PDS use of ISO 9660 Level 2 file names adheres to all the above restrictions, with the exception that the base name may be up to 27 characters long (total file name length not to exceed 31 characters). Thus, this format is sometimes referred to as the “27.3” format.

Note: In rare cases the following variations are allowed on the 27.3 format file name:

- The file name portion may be up to 29 characters long; or
- The extension may be up to 29 characters long.

In no case, however, may the total file name length, including the “.”, exceed 31 characters.

Preferred format:	BASENAME (1..27 characters) "." EXTENSION (3 characters)
Allowable format:	BASENAME (1..29 characters) "." EXTENSION (1..29 characters)
Actual format on archive medium:	BASENAME (1..29 characters) "." EXTENSION (1..29 characters) ";1"

Note that *only* the file and directory name specifications for Level 2 may be used in PDS archive volumes. All other Level 2 extensions are prohibited.

10.1.3 Specification for Files Delivered Electronically

Electronically delivered files must adhere to the ISO 9660 Level 2 specification with the exception that the base name of the file may be up to 36 characters long. Thus the total file name length, including a period and 3-character extension, is 40 characters. This format may be referred to as the "36.3" format. The limit of 40 characters is chosen because it is the maximum length of the value of the PRODUCT_ID keyword. As it is a common practice to use the file name as the unique product identifier, this limit will ensure that the file name is not too long to be a PRODUCT_ID.

Notes:

1. The 36.3 format exceeds the limit imposed by the ISO 9660:1988 Level 2 standard.
2. Directory names for electronic delivery must still follow the ISO 9660:1988 Level 2 standard, i.e., they are restricted to 31 characters as described in Section 8.2.
3. The total length of directory path and file name must not exceed 255 characters.
4. The 36.3 rule is specifically for electronic delivery. Files delivered on CD or DVD media must conform to the 27.3 rule.
5. Delivery of files on a "data brick", that is, a computer hard drive that can be mounted directly onto a computer or network, is considered an electronic delivery.

10.2 Reserved Directory Names, File Names and Extensions

A number of file names, directory names and file extensions are reserved for files that are required in PDS archive volumes under various circumstances. These reserved names and extensions are listed in the following sections for easy reference. For details concerning what directories and files are required where and when, see the indicated chapter.

10.2.1 Reserved Directory Names

The following directory names are reserved. The contents of these directories are described in Chapter 19, *Volume Organization and Naming*.

BROWSE
CALIB
CATALOG
DATA
DOCUMENT
EXTRAS
GAZETTER
GEOMETRY
INDEX
LABEL
SOFTWARE

10.2.2 Reserved File Names

The following file names are reserved. Not all of them are required in all cases. For a complete description of what files are required where and when, see Chapter 19, *Volume Organization and Naming*.

AAREADME.TXT	GAZINFO.TXT	PERSON.CAT
BROWINFO.TXT	GEOMINFO.TXT	REF.CAT
CALINFO.TXT	INDEX.TAB	SGIINFO.TXT
CATALOG.CAT	INDXINFO.TXT	SOFTINFO.TXT
CATINFO.TXT	INST.CAT	SUNINFO.TXT
CUMINDEX.TAB	INSTHOST.CAT	VOLDESC.CAT
DATASET.CAT	LABINFO.TXT	VOLDESC.SFD
DOCINFO.TXT	MACINFO.TXT	VOLINFO.TXT
ERRATA.TXT	MISSION.CAT	ZIPINFO.TXT
EXTRINFO.TXT	PCINFO.TXT	

10.2.3 Reserved Extensions

The file extensions listed in Table 10.1 are reserved. A brief description for each is provided in the table. Additional detail is contained in Chapter 19, *Volume Organization and Naming*, and Chapter 9, *Documents*.

Note that the presence of any given file extension in this table should in no way be construed as to imply that the associated format is acceptable for data archiving purposes. Please consult your Discipline Node for assistance in determining acceptable formats for that purpose.

10.3 Guidelines for Naming Sequential Files

In cases where file names are constructed from a time tag or sequential data object identifier, the following forms are suggested (but not required):

Pnnnnnnn.EXT

where ".EXT" is the file extension (see above) and P is a character indicating:

C *nnnnnnn* is a clock count value (e.g., "C3345678.IMG")
T *nnnnnnn* is a time value (e.g., "T870315.TAB")
F *nnnnnnn* is a frame ID or an image ID (e.g., "F242AO3.IMG")
N *nnnnnnn* is a numeric file identification number (e.g., "N003.TAB")

Extension	Description (use with files of this type)
ASC	Plain ASCII documentation files
BC	SPICE Binary format CK (pointing) files
BSP	SPICE Binary format SPK (ephemeris) files
CAT	Catalog object(s)
CSV	SPREADSHEET object(s)
DAT	Binary files (other than images)
DLL	Dynamic Link Library
DOC	Microsoft Word document
EPS	Encapsulated Postscript
EXE	Application or Executable
FIT	Image data with FITS (Flexible Image Transport System) header (preferred)
FMT	Include file for describing data object (meta data)
FTS	Image data with FITS (Flexible Image Transport System) header
GIF	GIF image
HTM *or* HTML	HTML document
IBG	Browse image data
IMG	Image data
IMQ	Image data that have been compressed (Not for use with JPEG 2000 compressed data.)
JP2	JPEG 2000 (JP2) formatted image
JPG	JPEG image
LBL	Detached label for describing data object
LIB	Library of object files
MAK	Makefile for compiling / linking application or executable
OBJ	Object file
PDF	Adobe PDF document
PNG	Portable Network Graphics
PS	Postscript
QUB	Spectral (or other) image QUBEs
RTF	Rich Text document
TAB	Tabular data, including ASCII TABLE objects with detached labels
TEX	TeX or LaTeX document
TI	SPICE Text IK (instrument parameters) files
TIF *or* TIFF	Tagged Image File Format documents
TF	SPICE Frames kernel files
TLS	SPICE Leap seconds kernel files
TPC	SPICE Physical and cartographic constants kernel files
TSC	SPICE Spacecraft clock coefficients kernel files
TXT	Plain text documentation files
XC	SPICE Transfer format CK (pointing) files
XES	SPICE E-kernel files
XSP	SPICE Transfer format SPK (ephemeris) files
ZIP	Zip-compressed files within PDS

Table 10.1 – Reserved File Extensions

(This page intentionally left blank.)

Chapter 11. Media Formats for Data Submission and Archive

This standard identifies the physical media formats to be used for data submission or delivery to the PDS or its science nodes. The PDS expects flight projects to deliver all archive products on magnetic or optical media or by electronic delivery, as negotiated with the science nodes.

Archive Planning - During archive planning, the data producer and PDS will determine the medium (or media) to use for data submission and archiving. This standard lists the media that are most commonly used for submitting data to and subsequently archiving data with the PDS. Delivery of data on media other than those listed here may be negotiated with the PDS on a case-by-case basis.

Physical Media for Archive - For archival products only media that conform to the appropriate International Standards Organization (ISO) standard for physical and logical recording formats may be used.

1. The preferred data delivery medium is the Compact Disk (CD-ROM or CD-Recordable) produced in ISO 9660 format, using Interchange Level 1, subject to the restrictions listed in Section 10.1.1.

2. Compact Disks may be produced in ISO 9660 format using Interchange Level 2, subject to the restrictions listed in Section 10.1.2.

3. Digital Versatile Disk (DVD-ROM or DVD-R) should be produced in UDF-Bridge format (Universal Disc Format) with ISO 9660 Level 1 or Level 2 compatibility.

Because of hardware compatibility and long-term stability issues, the use of 12-inch Write Once Read Many (WORM) disk, 8-mm Exabyte tape, 4-mm DAT tape, Bernoulli Disks, Zip disks, Syquest disks and Jaz disks is not recommended for archival use. WORM disk formats are proprietary to the specific vender hardware. Helical scan tape (8-mm or 4-mm) is prone to catastrophic read errors. Bernoulli, Zip, Jaz, Syquest and other vendor-specific storage media are prone to obsolescence.

Electronic Transfer and Storage of Archive – The ISO and UDF-Bridge standards mentioned above are specifically for CD-ROM, CD-R, DVD-ROM, and DVD-R media as noted. PDS recognizes that electronic delivery of archives is becoming as common a practice as delivery on physical media, and therefore is pursuing a more comprehensive standard. Pending the results of this investigation, the following restrictions apply to archival products delivered and stored electronically.

Notes:

1. File names are restricted to 40 characters total length, as described in Section 10.1.3. This exceeds the limit imposed by the ISO 9660:1988 standard.
2. Directory names are restricted to 31 characters total length, as described in Section 8.2

This is the same restriction as the ISO 9660:1988 standard.
3. The total length of directory path and file name must not exceed 255 characters.
4. Delivery of files on a “data brick”, that is, a computer hard drive that can be mounted directly onto a computer or network, is considered an electronic delivery.

11.1 CD-ROM Recommendations

11.1.1 Use of Variant Formats

The use of Extended Attribute Records (XARs), Rock Ridge Extensions or Macintosh Hybrid Disk Extensions on archival CD-ROMs is discouraged because these extensions can cause errors with CD-ROM drivers on some systems.

11.1.2 Premastering Recommendation

PDS recommends that CD-ROMs be premastered using a single-session, single-track format. Other formats have been found to be incompatible with some readers.

11.2 DVD Recommendations

11.2.1 Use of Variant Formats

The official volume structure for DVD media is UDF. DVD volumes should not be produced using ISO 9660 only. While current operating systems support ISO 9660 on DVD volumes, there is no guarantee that future operating system upgrades, set-top boxes or other new devices will continue to support ISO 9660 formatted DVD volumes.

11.2.2 Premastering Recommendation

PDS recommends that DVD-ROMs or DVD-Rs be premastered using a single-session, single-track format using the UDF-Bridge format.

11.2.3 Recommended DVD Formats

There are currently three "variants" of DVD media:

- DVD-5 - single sided, single layer (4.7 GB)
- DVD-9 - single sided, double layer (8.5 GB)
- DVD-10 - double sided, single layer (9.4 GB)

Currently, only the DVD-5 is approved by the PDS for archiving data. A waiver may be obtained for using the DVD-9 format if the archive consists of very large quantities of data (e.g., cost considerations may warrant using this format). The DVD-10 format is not recommended.

11.3 Packaging Software Files on a CD or DVD

The ISO 9660 Level 1 standard requires all pathnames and directory names to be in uppercase, and to be limited to eight characters with a three-character file extension for file names. In some cases it may be desirable to include software packages on an ISO 9660 Level 1 archive product that do not conform to these naming standards. The recommended method for packaging software is to use a "Zip" utility in accordance with the PDS standards for archiving data using Zip compression. See the *Zip Compression* chapter for more information.

11.4 Software Packaging Under Previous Versions of the Standard

Under previous versions of the Standards – prior to the adoption of the Zip standard (see the *Zip Compression* chapter) – archive products that included software specifically intended for the Mac and SUN operating systems used the following conventions:

1. Mac Software

   In this case the Mac files must be prepared in a particular format, as other platforms do not recognize the resource and data fork files that come with Mac applications. (For an example of properly formatted Mac software, see the NIHIMAGE software on the Magellan GxDR and Clementine EDR CD-ROMs.) The Mac utility "STUFFIT" is used to prepare the files by compressing them and encoding them using the BINHEX utility. Users will also need this STUFFIT utility in order to unpack the software for use. The procedure and software requirements should be described in a text file included on the CD-ROM (in the appropriate SOFTWARE/DOCUMENT subdirectory – see the *Volume Organization and Naming* chapter in this document).

   ***Example** – Text Documenting HQX Files*

```
                              Macintosh Software

This directory contains software that can be used to display the GXDR
images on a Macintosh II computer with an 8-bit color display.

NOTE:  Because of the way this CD-ROM was produced, it was not possible
to record this display program as a Macintosh executable file.  Anyone
who is unfamiliar with the Macintosh STUFFIT utility should contact the
PDS operator, 818-306-6026, SPAN address JPLPDS::PDS_OPERATOR, INTERNET
address PDS_OPERATOR@JPLPDS.JPL.NASA.GOV

The file IMAGE.HQX contains the NIH Image program, along with several
ancillary files and documentation in Microsoft WORD format.  It was
written by Wayne Rasband of the National Institutes of Health. The
program can be used to display any of the image files on the GXDR
CD-ROM disks.

The Image executable and manual are stored in BINHEX format, and the
utility STUFFIT or UNSTUFFIT must be used to:  1) decode the BINHEX
file IMAGE.HQX into IMAGE.SIT, using the 'DECODE BINHEX FILE...' option
in the Other menu; and 2) use 'OPEN ARCHIVE' from the File menu to
extract Image 1.40 from the STUFFIT archive file.  There are also
several other files in the archive file which should be unstuffed and
kept together in the same folder as the Image executable is stored.

The STUFFIT software is distributed as shareware.  STUFFIT, Version
```

```
1.5.1, is available by contacting:

 Raymond Lau                       MacNET:RayLau Usenet:raylau@dasys1.UUCP
 100-04 70 Ave.                    GEnie:RayLau
 Forest Hills, N.Y. 11375-5133     CIS:76174,2617
 United States of America.         Delphi:RaymondLau

Alternatively, STUFFIT CLASSIC, Version 1.6, is available by contacting:

 Aladdin Systems, Inc.
 Deer Park Center
 Suite 23A-171
 Aptos, CA 95003
 United States of America
```

2. SUN Software

The problem in this case is preserving the SUN file names, since case is significant in file names on that platform. Since the ISO standard requires all file and directory names to be uppercase, a disk premastered as an ISO CD may encounter problems in the case-sensitive SUN environment. Specifically, some CD readers mounted on SUN systems show file names as uppercase regardless of the format prior to mastering. If build routines ("make" files, for example) refer to lowercase file names, the corresponding files will not be found.

A method for dealing with this situation is to store the entire original directory structure and contents in a compressed, encoded archive (a compressed "tar" file, for example), and document the procedures and utilities needed to restore the files in the appropriate file. This is equivalent to the STUFFIT approach described above for Mac software.

Chapter 12. Object Description Language Specification and Usage

The following provides a complete specification for Object Description Language (ODL), the language used to encode data labels for the Planetary Data System (PDS) and other NASA data systems. This standard contains a formal definition of the grammar semantics of the language. PDS specific implementation notes and standards are referenced in separate sections.

12.1 About the ODL Specification

This standard describes Version 2.1 of ODL. Version 2.1 of ODL supersedes Versions 0 and 1 of the language, which were used previously by the PDS and other groups. For the most part, ODL Version 2.1 is backwardly compatible with previous versions of ODL. There are, however, some features found in ODL Versions 0 and 1 that have been removed from or changed within Version 2. The differences between ODL versions are described in Section 12.7.

Following is a sample ODL data label describing a file and its contents:

```
/* File Format and Length */
      RECORD_TYPE         = FIXED_LENGTH
      RECORD_BYTES        = 800
      FILE_RECORDS        = 860
/* Pointer to First Record of Major Objects in File */
      ^IMAGE              = 40
      ^IMAGE_HISTOGRAM    = 840
      ^ANCILLARY_TABLE    = 842
/* Image Description */
      SPACECRAFT_NAME     = VOYAGER_2
      TARGET_NAME         = IO
      IMAGE_ID            = "0514J2-00"
      IMAGE_TIME          = 1979-07-08T05:19:11Z
      INSTRUMENT_NAME     = NARROW_ANGLE_CAMERA
      EXPOSURE_DURATION   = 1.9200 <SECONDS>
      NOTE                = "Routine multispectral longitude
                            coverage, 1 of 7 frames"
/* Description of the Objects Contained in the File */
      OBJECT              = IMAGE
       LINES              = 800
       LINE_SAMPLES       = 800
       SAMPLE_TYPE        = UNSIGNED_INTEGER
       SAMPLE_BITS        = 8
      END_OBJECT          = IMAGE

      OBJECT              = IMAGE_HISTOGRAM
       ITEMS              = 25
       ITEM_TYPE          = INTEGER
       ITEM_BITS          = 32
      END_OBJECT          = IMAGE_HISTOGRAM

      OBJECT              = ANCILLARY_TABLE
       ^STRUCTURE         = "TABLE.FMT"
       END_OBJECT         = ANCILLARY_TABLE
      END
```

12.1.1 Implementing ODL

Notes to implementers of software to read and write ODL-encoded data descriptions appear throughout the following sections. These notes deal with issues beyond language syntax and semantics, but are addressed to assure that software for reading and writing ODL will be uniform. The PDS, which is the major user of ODL-encoded data labels, has imposed additional implementation requirements for software used within the PDS. These PDS requirements are discussed below where appropriate.

12.1.1.1 Language Subsets

Implementers are allowed to develop software to read or write subsets of the ODL. Specifically, software developers may opt to:

- Eliminate support for the GROUP statement (see Section 12.4.5.2 for additional information)
- Not support pointer statements
- Not support certain types of data values

For every syntactic element supported by an implementation, the corresponding semantics, as spelled out in this chapter, must be fully supported. Software developers should be careful to assure that language features will not be needed for their particular applications before eliminating them. Documentation on label reading/writing software should clearly indicate whether or not the software supports the entire ODL specification and, if not, should clearly indicate the features not supported.

12.1.1.2 Language Supersets

Software for writing ODL must not provide or allow lexical or syntactic elements over and above those described below. With the exception of the PVL-specific extensions below, software for reading ODL must not provide or allow any extensions to the language.

12.1.1.3 PDS Implementation of PVL-Specific Extensions

PDS implementation of software for reading ODL may, in some cases, provide handling of lexical elements that are included in the CCSDS specification of the Parameter Value Language (PVL), which is a superset of ODL. Extensions handled by such software include:

- BEGIN_OBJECT as a synonym for the reserved word OBJECT
- BEGIN_GROUP as a synonym for the reserved word GROUP
- Use of the semicolon (;) as a statement terminator

These lexical elements are not supported by software that writes the ODL subset. They must either be removed (in the case of semicolons) or replaced (in the case of the BEGIN_OBJECT and BEGIN_GROUP synonyms) upon output.

12.1.2 Notation

The formal description of the ODL grammar is given below in Backus-Naur Form (BNF). Language elements are defined using rules of the following form:

defined_element ::= definition

where the definition is composed from the following components:

1. Lower case words, some containing underscores, are used to denote syntactic categories. For example:

   units_expression

   Whenever the name of a syntactic category is used outside of the formal BNF specification, spaces take the place of underscores (for example, units expression).

2. Boldface type is used to denote reserved identifiers. For example:

   object

   Special characters used as syntactic elements also appear in boldface type.

3. Square brackets enclose optional elements. Elements within brackets occur zero or one times.

4. Square brackets followed immediately by an asterisk or plus sign specify repeated elements. In the case of an asterisk, the elements in brackets may appear zero, one, or more times. In the case of a plus sign, the elements in brackets must appear at least once. The repetitions occur from left to right.

5. A vertical bar separates alternative elements.

6. If the name of any syntactic category starts with an italicized part, it is equivalent to the category name without the italicized part. The italicized part is intended to convey some semantic information. For example, both *object*_identifier and *units*_identifier are equivalent to identifier; *object*_identifier is used in places where the name of an object is required and *units*_identifier is used where the name of some unit of measurement is expected.

12.2 Character Set

The character set of ODL is the International Standards Organization's ISO 646 character set. The U.S. version of the ISO 646 character set is ASCII; the ASCII graphical symbols are used throughout this document. In other countries certain symbols have a different graphical representation.

The ODL character set is partitioned into letters, digits, special characters, spacing characters, format effectors and other characters:

```
character :: =  letter | digit | special_character |
                spacing_character | format_effector |
                other_character
```

12.2.1 ODL Character Set - Letters

The letters are the uppercase letters A - Z and the lowercase letters a - z. ODL language elements are not case sensitive. Thus the following identifiers are equivalent:

- IMAGE_NUMBER
- Image_Number
- image_number

Case is significant inside of literal text strings, i.e., string “abc” is not the same as the string “ABC”.

12.2.2 ODL Character Set - Digits

The digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

12.2.3 ODL Character Set – Special Characters

The special characters used in ODL are:

Symbol	Name	Usage
=	Equals	The equals sign equates an attribute or pointer to a value.
{ }	Braces	Braces enclose an unordered set of values.
( )	Parentheses	Parentheses enclose an ordered sequence of values.
+	Plus	The plus sign indicates a positive numeric value.
-	Minus	The minus sign indicates a negative numeric value.
< >	Angle brackets	Angle brackets enclose a units expression associated with a numeric value.
.	Period	The period is the decimal place in real numbers.
"	Quotation Marks	Quotation marks denote the beginning and end of a text string value. Case is significant within the quotes of a text string.
’	Apostrophe	Apostrophes mark the beginning and end of a symbol value. Case is not significant within delimiting apostrophes (a.k.a. “single quotes”).
_	Underscore	The underscore separates words within an identifier.

,	Comma	The comma separates individual values in a set or sequence.
/	Slant	The slant character indicates division in units expressions. The slant is also part of the comment delimiter.
*	Asterisk	The asterisk indicates multiplication in units expressions. Two asterisks in a row indicate exponentiation in units expressions. The asterisk is also part of the comment delimiter.
:	Colon	The colon is used in attribute assignment statements to separate a namespace_identifier from an attribute_identifier (see Section 12.4.2). The colon separates hours, minutes and seconds within a time value.
#	Crosshatch	Also known as “the pound sign”, this symbol delimits the digits in an integer number value expressed in notation other than base-10.
&	Ampersand	The ampersand denotes continuation of a statement onto another line.
^	Circumflex	The circumflex (or caret) indicates that a value is to be interpreted as a pointer.

12.2.4 ODL Character Set – Spacing Characters

Two characters, called the spacing characters, separate lexical elements of the language and can be used to format characters on a line:

Space
Horizontal Tabulation

12.2.5 ODL Character Set – Format Effectors

The following ISO characters are format effectors, used to separate ODL encoded statements into lines:

Carriage Return
Line Feed
Form Feed
Vertical Tabulation

The spacing characters and format effectors are discussed further in section 12.4.1 below. There are other characters in the ISO 646 character set that are not required to write ODL statements and labels. These characters may, however, appear within text strings and quoted symbolic literals:

! $ % ; ? @ [ ] ` | ~

12.2.6 ODL Character Set – Control Characters

The category of other characters also includes the ASCII control characters except for horizontal tabulation, carriage return, line feed, form feed and vertical tabulation (e.g., the control characters that serve as spacing characters or format effectors). As with the printing characters in this category, the control characters in this category can appear within a text string. The handling of control characters within text strings and symbolic literals is discussed in Section 12.3.3 below.

12.3 Lexical Elements

This section describes the lexical elements of ODL. Lexical elements are the basic building blocks of the ODL. Statements in the language are composed by stringing lexical elements together according to the grammatical rules presented in Section 12.4. The lexical elements of ODL are:

- Numbers
- Dates and Times
- Strings
- Identifiers
- Special symbols used for operators, etc.

There is no inherent limit on the length of any lexical element. However, software for reading and writing ODL may impose limitations on the length of text strings, symbol strings and identifiers. It is recommended that at least 32 characters be allowed for symbol strings and identifiers and at least 400 characters for text strings.

12.3.1 Numbers

ODL can represent both integer numbers and real numbers. Integer numbers are usually represented in decimal notation (“123”), but ODL also provides for integer values in other number bases (for example, “2#1111011#” is the binary representation of the decimal integer “123”). Real numbers can be represented in simple decimal notation (“123.4”) or in scientific notation (i.e., with a base 10 exponent: “1.234E2”).

12.3.1.1 Integer Numbers In Decimal Notation

An integer number in decimal notation consists of a string of digits optionally preceded by a number sign. A number without an explicit sign is always taken as positive.

```
integer :: = [sign] unsigned_integer
unsigned_integer :: = [digit] +
sign ::=  + | -
```

***Examples** – Decimal Integers*

```
      0
    123
   +440
-150000
```

12.3.1.2 Integer Numbers In Based Notation

An integer number in based notation specifies the number base explicitly. The number base must be in the range 2 to 16, which allows for representations in the most popular number bases, including binary (base 2), octal (base 8) and hexadecimal (base 16). In general, for a number base X the digits 0 to X-1 are used. For example, in octal (base 8) the digits 0 to 7 are allowed. If X is greater than 10, then the letters A, B, C, D, E, F (or their lower case counterparts) are used as needed for the additional digits.

A based integer may optionally include a number sign. A number without an explicit sign is always taken as positive.

based_integer :: = radix **#** [sign] [extended_digit] + **#**
extended_digit :: = digit | letter
radix :: = unsigned_integer

***Examples** – Based Integers*

```
2#1001011#
    8#113#
    10#75#
    16#4B#
   16#+4B#
   16#-4B#
```

All but the last example above are equivalent to the decimal integer number 75. The final example is the hexadecimal representation of -75 decimal.

12.3.1.3 Real Numbers

Real numbers may be represented in floating-point notation ("123.4") or in scientific notation with a base 10 exponent ("1.234E2"). A real number may optionally include a sign. Unsigned numbers are always taken as positive.

real :: = [sign] unscaled_real | [sign] scaled_real
unscaled_real :: = unsigned_integer**.** [unsigned_integer] | **.**unsigned_integer
scaled_real :: = unscaled_real exponent
exponent :: = **E** integer | **e** integer

Note that the letter 'E' in the exponent of a real number may appear in either upper or lower case.

***Examples** – Real Numbers*

```
     0.0
   123.
+1234.56
     -.9981
    -1.E-3
31459e1
```

12.3.2 Dates and Times

ODL includes lexical elements for representing dates and times. The formats for dates and times are a subset of the formats defined by the International Standards Organization draft standard ISO/DIS 8601. (For information regarding PDS specific use of dates and times, see the *Date/Time* chapter in this document.)

12.3.2.1 Date and Time Values

Date and time scalar values represent a date, a time, or a combination of date and time:

date_time_value :: = date | time | date_time

The following rules apply to date values:

- The year must be Anno Domini. PDS requires a 4-digit year format be used (i.e., "2000", not "00").
- Month must be a number between 1 and 12.
- Day of month must be a number in the range 1 to 31, as appropriate for the particular month and year.
- Day of year must be in the range 1 to 365, or 366 in a leap year.

The following rules apply to time values:

- Hours must be in the range 0 to 23.
- Minutes must be in the range 0 to 59.
- Seconds, if specified, must be greater than or equal to 0 and less than 60.

The following rules apply to zone offsets within zoned time values:

- Hours must be in the range -12 to + 12 (the sign is mandatory).
- Minutes, if specified, must be in the range 0 to 59.

12.3.2.2 Implementation of Dates and Times

All ODL reading/writing software shall be able to handle any date within the 20th and 21st centuries. Software for writing ODL must always output full four-digit year numbers so that labels will be valid across century boundaries.

Times in ODL may be specified with unlimited precision, but the actual precision with which times will be handled by label reading/writing software is determined by the software implementers, based upon limitations of the hardware on which the software is implemented. Developers of label reading/writing software should document the precision to which times can be represented.

Software for writing ODL must not output local time values, since a label may be read in a time zone other than where it was written. Use either the UTC or zoned time format instead.

12.3.2.3 PDS Implementation of Dates and Times

PDS software for reading ODL labels interprets label times as UTC times. On output, a “Z” will be appended to label times.

12.3.2.4 Dates

Dates can be represented in two formats: as year and day of year; or as year, month and day of month.

```
date              :: = year_doy  | year_month_day
year_doy          :: = year-doy
year_month_day    :: = year-month-day
year              :: = unsigned_integer
month             :: = unsigned_integer
day               :: = unsigned_integer
doy               :: = unsigned_integer
```

Examples *– Dates*

```
1990-07-04
1990-158
2001-001
```

12.3.2.5 Times

Times are represented as hours, minutes and (optionally) seconds using a 24-hour clock. Times may be specified in Universal Time Coordinated (UTC) by following the time with the letter Z (for Zulu, a common designator for Greenwich Mean Time). Alternately, the time may be referenced to any time zone by following the time with a number that specifies the offset from UTC. Most time zones are an integral number of hours from Greenwich, but some are different by some non-integral time; both can be represented in the ODL. A time that is not followed by either the Zulu indicator or a time zone offset is assumed to be a local time.

```
time          :: = local_time | utc_time |  zoned_time
local time    :: = hour_min_sec
utc_time      :: = hour_min_sec Z
zoned_time    :: = hour_min_sec zone_offset
hour_min_sec  :: = hour: minute [:second]
zone_offset   :: = sign hour [: minute]
hour          :: = unsigned_integer
minute        :: = unsigned_integer
second        :: = unsigned_integer | unscaled_real
```

Note that either an integral or a fractional number of seconds can be specified in a time value.

***Examples** – Times*

```
12:00
15:24:12Z
01:10:39.4575+07  (time offset of 7 hours from UTC)
```

12.3.2.5.1 Combining Date and Time

A date and time can be specified together using the format below. Either of the two date formats can be combined with any time format - UTC, zoned or local.

date_time::=date **T** time

Note that, because this is a lexical element, spaces may not appear within a date, within a time or before or after the letter T.

***Examples** – Date/Times*

```
1990-07-04T12:00
1990-158T15:24:12Z
2001-001T01:10:39.457591+7
```

12.3.3 Strings

There are two kinds of string elements in ODL: text strings and symbol strings.

12.3.3.1 Text Strings

Text strings are used to hold arbitrary strings of characters.

```
quoted_text  ::=  "[character]*"
```

The empty string — a quoted text string with no characters within the delimiters — is allowed.

A quoted text string may not contain the quotation mark, which is reserved to be the text string delimiter. A quoted text string may contain format effectors, hence it may span multiple lines in a label: the lexical element begins with the opening quotation mark and extends to the closing quotation mark, even if the closing mark is on a following line. The rules for interpreting the characters within a text string, including format effectors, are given in the subsection on string values in Section 12.5.3.

12.3.3.2 Symbol Strings

Symbol strings are sequences of characters used to represent symbolic values. For example, an image ID may be a symbol string like 'J123-U2A', or a camera filter might be a symbol string like 'UV1'.

quoted_symbol ::= '[character]+'

A symbol string may not contain any of the following characters:

- The apostrophe, which is reserved to be the symbol string delimiter
- Format effectors, which means that a symbol string must fit on a single line
- Control characters

12.3.4 Identifiers

Identifiers are used as the names of objects, attributes and units of measurement. They can also appear as the value of a symbolic literal.

Identifiers are composed of letters, digits, and underscores. Underscores are used to separate words in an identifier. The first character of an identifier must be a letter. The last character may not be an underscore.

identifier : : = letter [letter | digit | _letter | _digit]*

Because ODL is not case sensitive, lower case characters in an identifier can be converted to their upper case equivalent upon input to simplify comparisons and parsing.

***Examples** – Identifiers*

VOYAGER
VOYAGER_2
BLUE_FILTER
USA_NASA_PDS_1_0007
SHOT_1_RANGE_TO_SURFACE

12.3.4.1 Reserved Identifiers

A few identifiers have special significance in ODL statements and are therefore reserved. They cannot be used for any other purpose (specifically, they may not be used to name objects or attributes):

end	end_group	end_object
group	object	begin_object

12.3.5 Special Characters

ODL is a simple language and it is usually clear where one lexical element ends and another begins. Spacing characters or format effectors may appear before a lexical element, between any pair of lexical elements, or after a lexical element without changing the meaning of a statement.

Some lexical elements incorporate special characters (e.g., the decimal point in real numbers or the quotation marks that delimit a text string). Some special characters are also lexical elements in their own right. These are:

=	The equals sign is the assignment operator.
,	The comma separates the elements of an array or a set.
*	The asterisk serves as the multiplication operator in units expressions.
/	The slant serves as the division operator within units expressions.
^	The circumflex denotes a pointer to an object.
<>	The angle brackets enclose units expressions.
()	The parentheses enclose the elements of a sequence.
{ }	The braces enclose the elements of a set.

The following two-character sequence is also a lexical element.

**	Two adjacent asterisks are the exponentiation sign within units expressions.

12.4 Statements

An ODL-encoded label is made up of a sequence of zero, one, or more statements followed by the reserve identifier **end**.

label ::= [statement]*
end

The body of a label is built from four types of statements:

```
statement :: = attribute_assignment_statement |
               pointer_statement |
               object_statement |
               group_statement
```

Each of the four types of statements is discussed below.

12.4.1 Lines and Records

Labels are also typically composed of lines, where each line is a string of characters terminated by a format effector or a string of adjacent format effectors. The following recommendations are given for how software that writes ODL should format a label into lines:

- There should be at most one statement on a line, although a statement may be more than a single line in length. As noted in Section 12.3.5 above, format effectors may appear before, after or between the lexical elements of a statement without changing the meaning of the statement. For example, the following statements are identical in meaning:

```
FILTER_NAME           =   {RED, GREEN, BLUE}

FILTER_NAME           =   {RED,
                           GREEN,
                           BLUE}
```

- Each line should end with a carriage return character followed immediately by a line feed character. This sequence is an end-of-line signal for most computer operating systems and text editors.
- The character immediately following the **END** statement must be either an optional spacing character or format effector, such as a space, line feed, carriage return, etc.

A line may include a comment. A comment begins with the two characters "/*" and ends with the two characters "*/". A comment may contain any character in the ODL character set except format effectors, which are reserved to mark the end of line (i.e., comments may not be more than one line long). Comments are ignored when parsing an ODL label. When the comment delimiters ("/*" and "*/") appear within a text string, they are not interpreted as a comment - they are simply part of the text string. For example, in the following example the comment will be included as part of the text string:

```
NOTE = "All good men come to the    /* Example of incorrect comment*/
        aid of their party"
```

Any characters on a line following a comment are ignored.

In some computer systems files are divided into records. Software for writing and reading ODL-encoded labels in record-oriented files should adhere to the following rules:

- A line of an ODL-encoded label may not cross a record boundary, i.e., each line should be contained within a single record. Any space left over at the end of a record after the last line in that record should be set to all space characters.
- The remainder of the record that contains the END statement is ignored. The data portion of the file begins with the next record in sequence.

12.4.2 Attribute Assignment Statement

The attribute assignment statement is the most common type of statement in ODL and is used to specify the value for an attribute of an object. The value may be a singular scalar value, an ordered sequence of values, or an unordered set of values. In PDS ODL files, if there are multiple values, a single attribute assignment statement must be used with either sequence or set syntax; no assignment statement may be repeated.

The attribute assignment statement may optionally contain a namespace_identifier. When a namespace_identifier is prepended to the element_identifier statement, it indicates that the element_identifier has a local definition within the context indicated by the namespace_identifier.

assignment_statement ::= *attribute*_identifier = value

where *attribute*_identifier::= *element*_identifier |
*namespace*_identifier:*element*_identifier

The syntax and semantics of values are given in Section 12.5.

Examples *– Assignment Statements*

```
RECORD_BYTES                = 800
TARGET_NAME                 = JUPITER
SOLAR_LATITUDE              = (0.25 <DEG>, 3.00 <DEG>)
FILTER_NAME                 = {RED,
                               GREEN,
                               BLUE}
```

Examples *– Assignment Statements that use namespace_identifier*

```
CASSINI:TARGET_NAME = JUPITER
MRO:SOLAR_LATITUDE  = (0.25 <DEG>, 3.00 <DEG>)
VOYAGER:FILTER_NAME = { RED, GREEN, BLUE }
```

12.4.3 Pointer Statement

The pointer statement indicates the location of an object.

pointer_statement :: = ^*object*_identifier = value

As with the attribute assignment statement, the value may be a scalar value, an ordered sequence of values, or an unordered set of values.

A common use of pointer statements is to reference a file containing an auxiliary label. For example:

```
^STRUCTURE = "TABLE.FMT"
```

This is a pointer statement pointing to a file named "TABLE.FMT" that contains a description of the structure of the ancillary table from our sample label. Another use of the pointer statement is to indicate the position of an object within another object. This is often used to indicate the position of major objects within a file. The following examples are from the sample label in Section 12.1:

```
^IMAGE               = 40
^IMAGE_HISTOGRAM     = 840
^ANCILLARY_TABLE     = 842
```

The first pointer statement above indicates that the image is located starting at the 40th record from the beginning of the present file. If an integer value is used to indicate the relative position of an object, the units of measurement of position are determined by the nature of the object. For files, the default unit of measurement is records. Alternatively, a units expression can be specified for the integer value to indicate explicitly the units of measurement for the position. For example, this pointer:

```
^IMAGE         = 10200 <BYTES>
```

indicates that the image starts 10,200 bytes from the beginning of the file.
The object pointers above reference locations in the same files as the label containing the pointer. Pointers may also reference either byte or record locations in data files that are detached, or separate, from the label file:

```
^IMAGE         = ("IMAGE.DAT", 10)
^HEADER        = ("IMAGE.DAT", 512 <BYTES>)
```

12.4.4 OBJECT Statement

The OBJECT statement begins the description of an object. The description typically consists of a set of attribute assignment statements defining the values of the object's attributes. If an object is itself composed of other objects, then OBJECT statements for the component objects are nested within the object's description. There is no limit to the depth to which OBJECT statements may be nested.

The format of the OBJECT statement is:

object_statement ::= **object** = *object*_identifier

[statement]*
end_object [= *object*_identifier]

The object identifier gives a name to the particular object being described. For example, in a file containing images of several planets, the image object descriptions might be named VENUS_IMAGE, JUPITER_IMAGE, etc. The object identifier at the end of the OBJECT statement is optional, but if it appears it must match the name given at the beginning of the OBJECT statement.

12.4.4.1 Implementation of OBJECT Statements

It is recommended that all software for writing ODL include the object identifier at the end as well as the beginning of every OBJECT statement.

12.4.5 GROUP Statement

The GROUP statement is used to group together statements that are not components of a larger object. For example, in a file containing many images, the group BEST_IMAGES might contain the object descriptions of the three highest quality images. The three image objects in the BEST_IMAGES group don't form a larger object: all they have in common is their superior quality.

The GROUP statement is also used to group related attributes of an object. For example, if two attributes of an image object are the time at which the camera shutter opened and closed, then the two attributes might be grouped as follows:

```
GROUP      = SHUTTER_TIMES
  START    = 12:30:42.177
  STOP     = 14:01:29.265
END_GROUP  = SHUTTER_TIMES
```

The format of the group statement is as follows:

group_statement ::= **group** = *group*_identifier
[statement]*
end_group [= *group*_identifier]

The group identifier gives a name to the particular group, as shown in the example for shutter times above. The object identifier at the end of the GROUP statement is optional, but if it appears it must match the name given at the beginning of the GROUP statement. Groups may be nested within other groups. There is no limit to the depth to which groups can be nested.

As opposed to the above ODL implementation, the PDS applies the following restrictions to the use of GROUPS:

1. The GROUP structure may only be used in a data product label which also contains one or more data OBJECT definitions.

2. The GROUP statement must contain only attribute assignment statements, include pointers, or related information pointers (i.e., no data location pointers). If there are multiple values, a single statement must be used with either sequence or set syntax; no attribute assignment statement or pointer may be repeated.
3. GROUP statements may not be nested.
4. GROUP statements may not contain OBJECT definitions.
5. Only PSDD elements may appear within a GROUP statement.
6. The keyword contents associated with a specific GROUP identifier must be identical across all labels of a single data set (with the exception of the "PARAMETERS" GROUP, as explained .

Use of the GROUP structure must be coordinated with the responsible PDS discipline Node.

12.4.5.1 Implementation of GROUP Statements

It is recommended that all software for writing ODL include the group identifier at the end as well as the beginning of every GROUP statement.

12.4.5.2 PDS Usage of GROUP

Although ODL includes the GROUP statement, the PDS does not recommend its use because of confusion concerning the difference between OBJECT and GROUP.

12.5 Values

ODL provides scalar values, ordered sequences of values, and unordered sets of values.

value :: = scalar_value | sequence_value | set_value

A scalar value consists of a single lexical element:

scalar_value :: = numeric_value |
date_time_value |
text_string_value |
symbol_value

The format and use of each of these scalar values are discussed in the sections below.

12.5.1 Numeric Values

A numeric scalar value is either a decimal or based integer number, or a real number. A numeric scalar value may optionally include a units expression.

numeric_value :: = integer [units_expression] |
based_integer [units_expression] |
real [units_expression]

12.5.2 Units Expressions

Many of the values encountered in scientific data are measurements of something. In most computer languages, only the magnitude of a measurement is represented, without the units of measurement. ODL, however, can represent both the magnitude and the units of a measurement. A units expression has the following format:

units_expression	:: = < units_factor [mult_op units_factor] * >
units_factor	:: = *units*_identifier [exp_op integer]
mult_op	:: = * | /
exp_op	:: = **

A units expression is always enclosed within angle brackets. The expression may consist of a single units identifier like "KM",for kilometers, or "SEC", for seconds (for example, "1.341E6 <KM>" or "1.024 <SEC>"). More complex units can also be represented; for example, the velocity "3.471 <KM/SEC>" or the acceleration "0.414 < KM/SEC/SEC>". There is often more than one way to represent a unit of measure. For example:

```
0.414          <KM/SEC/SEC>

0.414          <KM/SEC**2>

0.414          <KM*SEC**-2>
```

are all valid representations of the same acceleration. The following rules apply to units expressions:

- The exponentiation operator can specify only a decimal integer exponent. The exponent value may be negative, which signifies the reciprocal of the units. For example, "60.15 <HZ>" and "60.15 <SEC**-1>" are both ways to specify a frequency.
- Individual units may appear in any order. For example, a force might be specified as either "1.55 <GM*CM/ SEC**2>" or "1.55 <CM*GM/SEC**2>".

12.5.2.1 Implementation of Numeric Values

There is no defined maximum or minimum magnitude or precision for numeric values. In general, the actual range and precision of numbers that can be represented will be different for each kind of computer used to read or write an ODL-encoded label. Developers of software for reading/writing ODL should document the following:

- The largest magnitude positive and negative integers that can be represented
- The largest magnitude positive and negative real numbers that can be represented
- The minimum number of significant digits that a real number can be guaranteed to have without loss of precision. This is to account for the loss of precision that can occur when representing real numbers in floating point format within a computer. For example, a 32-bit floating-point number with 24 bits for the mantissa can guarantee at most 6 significant digits will be exact (the seventh and subsequent digits may not be exact because of truncation and round-off errors).

If software for reading ODL encounters a numeric value too large to be represented, the software must report an error to the user.

12.5.3 Text String Values

A text string value consists of a text string lexical element:

text_string_value :: = quoted_text

12.5.3.1 Implementation of String Values

A text string read in from a label is reassembled into a string of characters. The way in which the string is broken into lines in a label does not affect the format of the string after it has been reassembled. The following rules are used when reading text strings:

- If a format effector or a sequence of format effectors is encountered within a text string, the effector (or sequence of effectors) is replaced by a single space character, unless the last character is a hyphen (dash) character. Any spacing characters at the end of the line are removed and any spacing characters at the beginning of the following line are removed. This allows a text string in a label to appear with the left and right margins set at arbitrary points without changing the string value. For example, the following two strings are the same:

  “To be or not to be”

  and

  “To be or
  not to be"

- If the last character on a line prior to a format effector is a hyphen (dash) character, the hyphen is removed with any spacing characters at the beginning of the following line. This follows the standard convention in English of using a hyphen to break a word across lines. For example, the following two strings are the same:

  “The planet Jupiter is very big”

  and

  “The planet Jupi-
  ter is very big”

- Control codes, other than the horizontal tabulation character and format effectors, appearing within a text string are removed.

12.5.3.1.1 PDS Text String Formatting Conventions

The PDS defines a set of format specifiers that can be used in text strings to indicate the formatting of the string on output. These specifiers can be used to indicate where explicit line breaks should be placed, and so on. The format specifiers are:

\n	Indicates that an end-of-line sequence should be inserted.
\t	Indicates that a horizontal tab character should be inserted.
\f	Indicates that a page break should be inserted.
\v	Must be used in pairs, begin and end. Interpreted as verbatim.
\\	Used to place a backslash in a text string.

For example, the string

“This is the first line \n and this is the second line.”

will print as:

This is the first line
and this is the second line.

Note: These format specifiers have meaning only when a text string is printed - not when the string is read in or stored.

12.5.4 Symbolic Literal Values

A symbolic value may be specified as either an identifier or a symbol string:

symbolic-value :: = identifier | quoted_symbol

The following statements assign attributes to symbolic values specified by identifiers:

```
TARGET_NAME       =  IO
SPACECRAFT_NAME   =  VOYAGER_2
SPACECRAFT_NAME   =  'VOYAGER-2'
SPACECRAFT_NAME   =  'VOYAGER 2'
REFERENCE_KEY_ID  =  SMITH1997
REFERENCE_KEY_ID  =  'LAUREL&HARDY1997'
```

The quotes must be used if the symbolic value does not have the proper format for an identifier or if it contains characters not allowed in an identifier. For example, the value ‘FILTER_+_7’ must be enclosed within quotes, since this would not be a legal ODL identifier. Similarly, the symbolic value ‘U13-A4B’ must be in quotes because it contains a special character (the dash) not allowed in an identifier. There is no harm in putting a legal identifier within quotes. For example:

```
SPACECRAFT_NAME = 'VOYAGER_2'
```

is equivalent to the second example in the list above.

Symbolic values may not contain format effectors, i.e., they may not cross a line boundary.

12.5.4.1 Implementation of Symbolic Literal Values

Symbolic values are converted to upper case on input. This means that a lowercase string is converted to the equivalent uppercase string; as in the following example:

Original string:	`SPACECRAFT_NAME = 'Voyager_2'`
Converted string:	`SPACECRAFT_NAME = 'VOYAGER_2'`

12.5.4.2 PDS Convention for Symbolic Literal Values

Since the current use of the ODL within the PDS does not require syntactic differentiation between symbols and text strings, PDS prefers that double quotation marks (") be used instead of apostrophes around symbol strings.

12.5.5 Sequences

A sequence represents an ordered set of values. It can be used to represent arrays and other kinds of ordered data. Only one- and two-dimensional sequences are allowed.

sequence_value	:: = sequence_1D | sequence_2D
sequence_1D	:: = **(**scalar_value [**,** scalar_value]***)**
sequence_2D	:: = **(**sequence_1D [, sequence_1D]***)**

A sequence may have any kind of scalar value for its members. It is not required that all the members of the sequence be of the same type. Thus a sequence may represent a heterogeneous record. Each member of a two-dimensional sequence is a one-dimensional sequence. This can be used, for example, to represent a table of values. The order in which members of a sequence appear must be preserved. There is no upper limit on the number of values in a sequence.

Example of a 1D sequence:

AVERAGE_ECCENTRICITY = (0.2056, 0.0068, 0.0167, 0.0934, 0.0483, 0.0560)

Example of a 2D sequence:

MRO:LOOKUP_CONVERSION_TABLE = ((0, 1008), (1009, 1025), (1026, 1043), …)

12.5.6 Sets

Sets are used to specify unordered values drawn from some finite set of values.

set_value :: = {scalar_value [, scalar_value]*} | {}

Note that the empty set is allowed: The empty set is denoted by opening and closing brackets with nothing except optional spacing characters or format effectors between them.

The order in which the members appear in the set is not significant and the order need not be preserved when a set is read and manipulated. There is no upper limit on the number of values in a set.

Example

```
FILTER_NAME  = { RED, BLUE, GREEN, HAZEL }
```

12.5.6.1 PDS Restrictions on Sets

The PDS allows only symbol values and integer values within sets.

12.6 ODL Summary

Character Set (Section 12.2)

ODL uses the ISO 646 character set (the American version of the ISO 646 standard is ASCII). The ODL character set is partitioned as follows:

```
character              : : = letter | digit | special_character |
                                spacing_character | format_effector |
                                other_character
letter                 : : = A-Z | a-z
digit                  : : =  0 | 1  | 2 | 3 | 4 | 5 | 6 | 7 | 8| 9
special_character      : : =  { | } | ( | ) | + | - | . | " | ' | = |
                                 _ | , | / | * | : | # | & | ^ | < | >
spacing_character      : : = space | horizontal tabulation
format_effector        : : = carriage return | line feed |
                                 form feed | vertical tabulation
other_character        : : =  ! | $ | % | ; | ? | @ | [ | ] | ` | ~ |
                                vertical bar | other control characters
```

Lexical Elements (Section 12.3)

```
integer                : : = [sign] unsigned_integer
unsigned_integer       : : = [digit]+
sign                   : : = + | -
based_integer          : : = radix # [sign] [extended_digit]+ #
```

extended_digit	: : = digit | letter
radix	: : = unsigned_integer
real	: : = [sign] unscaled_real | [sign] scaled_real
unscaled_real	: : = unsigned_integer **.** [unsigned_integer] | **.** unsigned_integer
scaled_real	: : = unscaled_real exponent
exponent	: : = **E** integer | **e** integer
date	: : = year_doy | year_month_day
year_doy	: : = year **-** doy
year_month_day	: : = year **-** month **-** day
year	: : = unsigned_integer
month	: : = unsigned_integer
day	: : = unsigned_integer
doy	: : = unsigned_integer
time	: : = local_time | utc_time | zoned_time
local_time	: : = hour_min_sec
utc_time	: : = hour_min_sec **Z**
zoned_time	: : = hour_min_sec zone_offset
hour_min_sec	: : = hour **:** minute [**:** second]
zone_offset	: : = sign hour [**:** minute]
hour	: : = unsigned_integer
minute	: : = unsigned_integer
second	: : = unsigned_integer | unscaled_real
date_time	: : = date **T** time
quoted_text	: : = “[character]*”
quoted_symbol	: : = ‘[character]+’
identifier	: : = letter [letter | digit | _letter | **_**digit]*

Statements (Section 12.4)

label	: : = [statement]* **end**
statement	: : = assignment_stmt | pointer_stmt | object_stmt | group_stmt
assignment_stmt	: : = element_identifier = value | namespace_identifier:element_identifier = value
pointer_stmt	: : = ^*object*_identifier **=** value
object_stmt	: : = **object =** *object*_identifier [statement]* **end_object** [= *object*_identifier]
group_stmt	: : = **group =** *group*_identifier [statement]* **end_group** [= *group*_identifier]

Values (Section 12.5)

```
value               : : = scalar_value | sequence_value |   set_value
scalar_value        : : = numeric_value | date_time_value |
                              text_string_value | symbolic_value
numeric_value       : : = integer [units_expression] |
                              based_integer [units_expression] |
                              real [units_expression]
units_expression    : : =<units_factor[mult_op units_factor]* >
units_factor        : : = units_identifier [exp_op integer]
mult_op             : : = * | /
exp_op              : : = **
date_time_value     : : = date | time  | date_time
text_string_value   : : = quoted_text
symbolic_value      : : = identifier | quoted_symbol
sequence_value      : : =sequence_lD | sequence_2D
sequence_1D         : : = (scalar_value [, scalar_value]*)
sequence_2D         : : = ([sequence_lD]+)
set_value           : : = { scalar_value [,scalar_value]* } | { }
```

12.7 Differences Between ODL Versions

This section summarizes the differences between the current Version 2 of ODL and the previous Versions 0 and 1. Software can be constructed to read all three versions of ODL, however it is important that software for writing labels only write labels that conform to ODL Version 2.

12.7.1 Differences from ODL Version 1

Version 1 labels were used on the *Voyager to the Outer Planets* CD-ROM disks and many other data sets. Version 1 did not include the GROUP statement and had more restrictive definitions for sets, which were limited to integer or symbolic literal values, and sequences, which were limited to arrays of homogeneous values. The following sections detail non-compatible differences and how they can be handled by software writers.

12.7.1.1 Ranges

Version 1 of ODL had a specific notation for integer ranges:

range_value :: = integer..integer

This notation is not allowed in ODL Version 2, though parsers may still recognize the ‘double-dot’ range notation. On output, a range is now encoded as a two value sequence, with the low-value of the range being the first element of the sequence and the high-value being the second element of the sequence.

12.7.1.1.1 Delimiters in Sequences and Sets

In Version 1 the individual values in sets and sequences could be separated by a comma or by a spacing character. As of Version 2, a comma is required. Parsers may allow spacing characters between values rather than commas. Software that writes ODL should place commas between all values in a sequence or set.

12.7.1.1.2 Exponentiation Operator in Units Expressions

In Version 1 of ODL the circumflex character (^) was used as the exponentiation operator in units expressions rather than the two-asterisk sequence (**). Parsers may still allow the circumflex to appear within units expressions as an exponentiation operator. Software for writing ODL should use only the ** notation.

12.7.2 Differences from ODL Version 0

Version 0 of ODL was developed for and used on the *PDS Space Science Sampler* CD-ROM disks. The major difference between this and subsequent versions is that Version 0 did not include the OBJECT statement. All of the attributes specified in a label described a single object: the file that contained the label (or that was referenced by a pointer).

12.7.2.1 Date–Time Format

ODL Version 0 was produced prior to the space community's acceptance of the ISO/DIS 8601 standard for dates and time and it uses a different date and date-time format. The format for Version 0 dates and date-times is as follows:

```
date        :: = year / month / day_of_month  | year / day_of_year
date_time   :: = date - time zone
zone        :: = < identifier>
```

The options for time specification in ODL Version 0 are a subset of those in Version 2. Consequently, parsers that handle Version 2 time formats will also handle Version 0 times.

12.7.3 ODL/PVL Usage

The concept for a Parameter Value Language/Format (PVL) is being formalized by the Consultative Committee for Space Data Systems (CCSDS). It is intended to provide a human readable data element/value structure to encode data for interchange. The CCSDS version of the PVL specification is in preliminary form.

Some organizations that deal with the PDS have accepted PVL as their standard language for product labels. PVL is a superset of ODL, so some PVL constructs are not supported by the PDS. In addition, some ODL constructs may be interpreted differently by PVL software.

The ODL/PVL usage standard defines restrictions on the use of ODL/PVL in archive quality data sets. These restrictions are intended to ensure the compatibility of PVL with ODL and existing software.

1. A label constructed using PVL may be attached - embedded in the same file as the data object it describes, or detached - residing in a separate file and pointing to the data file the label describes.

2. All statements must be terminated by a <CR> <LF> pair. Semicolons may not be used to terminate statements.

3. Only alphanumeric characters and the underscore character may be used in data elements and undelimited text values (literals). In addition, data elements and undelimited text values must begin with a letter.

4. Keywords must be 30 characters or less in length.

5. Keywords and standard values must be in upper case. Literals and strings may be in upper case, lower case, or mixed case.

6. Comments must be contained on a single line, and a comment terminator (*/) must be used. Comments may not be embedded within statements. Comments may not be used on the same line as any statement if the comment precedes the statement. Comments may be on the same line as a statement if the comment follows the statement and is separated from the statement by at least one white space, but this is not recommended.

7. Text values that cross line boundaries must be enclosed in double quotation marks (“ ”).

8. Values that consist only of letters, numbers, and underscores and that begin with a letter may be used without quotation marks. All other text values must be enclosed in either single (‘ ’) or double (“ ”) quotation marks.

9. Sequences are limited to two dimensions. Null (empty) sequences are not allowed. Sets are limited to one dimension. In other words, sets and sequences may not be used inside a set.

10. Only the OBJECT, END_OBJECT, GROUP and END_GROUP aggregation markers may be used.

11. Unit expressions are only allowed following numeric values (i.e., “DATA_ELEMENT = 7 <BYTES>” is valid. but “DATA_ELEMENT = MANY <METERS>” is not).

12. Unit expressions may include only alphanumeric characters, the underscore, and the symbols “*”, “/”, “(”, “)”, and “**” (the last representing exponentiation).

13. Signs may not be used in non-decimal numbers (i.e., “2#10001#” is valid, but “-2#10001#” and “2#-10001#” are not). Only the bases 2, 8, and 16 may be used for non-decimal numbers.

14. Alternate time zones (e.g., YYYY-MM-DDTHH:MM:SS.SSS + HH:MM) may not be used in a PDS label. The only allowed time formats are

    (1) YYYY-MM-DDTHH:MM:SS.SSS.
    (2) YYYY-DDDTHH:MM:SS.SSS.

    See Section 7.3.2(6) for a more detailed description.

15. Values in integral parts of dates and times must be padded on the left with zeroes as necessary to fill the field. In other words, the first of April in the year 2001 must be written as “2001-04-01”, *not* “2001-4-1”

16. An END statement must conclude each ODL/PVL statement list.

The following are guidelines for formatting ODL/PVL expressions.

1. The assignment symbol (=) must be surrounded by blanks.

2. Assignment symbols (=) should be aligned if possible.

3. Keywords placed inside an aggregator (OBJECT or GROUP) must be indented with respect to the OBJECT and END_OBJECT or GROUP and END_GROUP statements which enclose them.

4. PDS label lines must be 80 characters or less in length, including the end-of-statement (i.e., <CR> <LF>) delimiter. (Note that while 80 characters can be displayed on most screens, some editors and databases will wrap or truncate lines that exceed 72 characters.)

5. Horizontal tab characters may not be used in PDS labels. Although both ODL and PVL allow the use of these characters some simple parsers cannot handle them. The equivalent number of space characters should be used instead.

(This page intentionally left blank.)

Chapter 13. PDS Objects / Groups

The Planetary Data System has designed a set of standard Objects and Groups to be used for submitting catalog object information as well as for labeling data products. These standard Objects and Groups, along with definitions of individual keywords comprising those Objects and Groups, are defined in the *Planetary Science Data Dictionary*. In addition, Object and Group definitions and examples are also included in Appendix A and Appendix B of this document.

13.1 Generic and Specific Data Object Definitions

For each type of data object that PDS has defined (i.e., IMAGE, TABLE, etc.), there are two categories of definitions: generic and specific. A *generic object definition* is the universal definition of an object, or superset of keywords that can be used. A *specific object definition* is a subset of keywords used for a particular data product to allow effective use of validation tools.

Generic object definitions are designed and approved by the Planetary Data System, and defined in the *Planetary Science Data Dictionary*. Each object definition lists the elements and sub-objects required to be present each time the object is used in a product label. The dictionary definition also provides a list of additional, optional keywords that are frequently used by data preparers. Finally, note that any element defined in the PSDD may be included as an optional element in any object definition, at the discretion of the data preparer.

A specific object definition is defined for a particular data product and is based on a single generic object. The data preparer, in consultation with a data engineer, combines all the required elements of that object with a set of optional elements selected for their relevance to the data at hand. The result is a specific object definition. This definition is subject to approval during a design review.

The following examples illustrate the evolution from the generic IMAGE object to a specific IMAGE object, followed by an instance of that specific IMAGE. Note that when a specific object definition is created and used, the usage must be consistent for all labels using that object.

```
OBJECT                     = GENERIC_OBJECT_DEFINITION
NAME                       = IMAGE
STATUS_TYPE                = APPROVED
STATUS_NOTE                = "V2.1  1991-01-20  MDM  New Data Object Definition"
DESCRIPTION                = "An image object is a regular array of sample
values.  Image objects are normally processed with special display tools to
produce a visual representation of the sample values. This is done by assigning
brightness levels or display colors to the various sample values. Images are
composed of LINES   and SAMPLES.  They may contain multiple bands, in one of
several storage orders.

Note:  Additional engineering values may be prepended or appended to each LINE
of an image, and are stored as concatenated TABLE objects, which must be named
LINE_PREFIX and LINE_SUFFIX.  IMAGE objects may be associated with other
objects, including HISTOGRAMs, PALETTEs, HISTORYs and TABLEs which contain
statistics, display parameters, engineering values or other ancillary data."

SOURCE_NAME                = "PDS CN/M.MARTIN"
REQUIRED_ELEMENT_SET         =                               {LINE_SAMPLES,
```

```
LINES, SAMPLE_BITS,
                             SAMPLE_TYPE}
OPTIONAL_ELEMENT_SET        =                               {BAND_SEQUENCE,
BAND_STORAGE_TYPE,
                            BANDS, CHECKSUM, DERIVED_MAXIMUM,
                            DERIVED_MINIMUM, DESCRIPTION,
                            ENCODING_TYPE, FIRST_LINE,
                            FIRST_LINE_SAMPLE, INVALID,
                            LINE_PREFIX_BYTES, LINE_SUFFIX_BYTES, MISSING,
                            OFFSET, SAMPLE_BIT_MASK, SAMPLING_FACTOR,
                            SCALING_FACTOR, SOURCE_FILE_NAME,
                            SOURCE_LINES,  SOURCE_LINE_SAMPLES,
                            SOURCE_SAMPLE_BITS, STRETCHED_FLAG,
                            STRETCH_MAXIMUM, STRETCH_MINIMUM, PSDD}
REQUIRED_OBJECT_SET        = "N/A"
OPTIONAL_OBJECT_SET        = "N/A"

OBJECT_CLASSIFICATION_TYPE = STRUCTURE

OBJECT                     = ALIAS
NAME                       = "N/A"
USAGE_NOTE                 = "N/A"
END_OBJECT                 = ALIAS

END_OBJECT                 = GENERIC_OBJECT_DEFINITION
```

This next example illustrates an IMAGE object definition being used for a specific case.

```
OBJECT                     = SPECIFIC_OBJECT_DEFINITION
NAME                       = XYZ_IMAGE
STATUS_TYPE                = APPROVED
STATUS_NOTE                = "V2.1 1991-02-10  TMA New specific data object
                              definition"
DESCRIPTION                = "The XYZ image is..."

SOURCE_NAME                = "PDS CN/M.MARTIN"
REQUIRED_ELEMENT_SET       = {LINE_SAMPLES, LINES, SAMPLE_BITS,
                               SAMPLE_TYPE, SAMPLING_FACTOR,
                               SOURCE_FILE_NAME,
                               SOURCE_LINES,  SOURCE_LINE_SAMPLES,
                               SOURCE_SAMPLE_BITS, FIRST_LINE,
                               FIRST_LINE_SAMPLE}

OBJECT_CLASSIFICATION_TYPE = STRUCTURE

OBJECT                     = ALIAS
NAME                       = "N/A"
USAGE_NOTE                 = "N/A"
END_OBJECT                 = ALIAS

END_OBJECT                 = SPECIFIC_ OBJECT_DEFINITION
```

13.1.1 Primitive Objects

Generic objects have a subclass called primitive objects that includes the ARRAY, COLLECTION, ELEMENT, and BIT_ELEMENT objects. The primitive objects are used as the building blocks for describing very irregular data that cannot be accommodated by any other generic object. If at all possible, standard, well-supported generic objects (such as TABLE and IMAGE) should be used to describe archival data.

13.2 Generic and Specific Data Group Definitions

For each type of data Group that PDS has defined (i.e., PARAMETERS, etc.), there are two categories of definitions: generic and specific. A *generic group definition* is the universal definition of a group, or superset of keywords that can be used. A *specific group definition* is a subset of keywords used for a particular data product to allow effective use of validation tools.

As with OBJECTs (see PDS Standards Reference, section 13.1), there are two categories of GROUPs, generic and specific. The generic GROUP is the universal definition of the GROUP, specified in an appendix of the Standards Reference. The specific GROUP is an implementation of the generic GROUP for a particular data set. Shown below is a generic GROUP definition, and then an example of an instance of that GROUP in a data product.

```
OBJECT                     = GENERIC_GROUP_DEFINITION
NAME                       = CAMERA_MODEL
STATUS_TYPE                = PENDING
STATUS_NOTE                = "V1.0  2001-07-09 EDR  New Group Definition"
DESCRIPTION                = "A camera model group is a collection of parameters
necessary to fully describe the geometric characteristics of a camera system."

SOURCE_NAME                = "PDS IMG/E. RYE"
REQUIRED_ELEMENT_SET         =                              {CAMERA_MODEL_NAME,
                           CAMERA_MODEL_TYPE,
                              CAMERA_MODEL_DESC, CALIBRATION_SOURCE_ID,
                              GEOMETRY_SOURCE_ID, COORDINATE_SYSTEM_NAME,
                              MODEL_COMPONENT_ID, MODEL_COMPONENT_NAME,
                              MODEL_COMPONENT_UNIT_ID}
OPTIONAL_ELEMENT_SET =       {MODEL_COMPONENT_1_VECTOR,
                              MODEL_COMPONENT_2_VECTOR,
                              MODEL_COMPONENT_3_VECTOR,
                              MODEL_COMPONENT_4_VECTOR,
                              MODEL_COMPONENT_5_VECTOR,
                              MODEL_COMPONENT_6_VECTOR,
                              MODEL_COMPONENT_7_VECTOR, PSDD}

OBJECT                     = ALIAS
NAME                       = "N/A"
USAGE_NOTE                 = "N/A"
END_OBJECT                 = ALIAS

END_OBJECT                 = GENERIC_GROUP_DEFINITION
```

An example of using a GROUP follows:

```
GROUP                      = CAMERA_MODEL
CAMERA_MODEL_NAME          = "MIPS-0"
CAMERA_MODEL_TYPE          = "CAHV"
^CAMERA_MODEL_DESC         = "CAHV.ASC"
CALIBRATION_SOURCE_ID      = "UOFA-BACKLASH"
GEOMETRY_SOURCE_ID         = "TELEMETRY"
COORDINATE_SYSTEM_NAME     = "IMP-CAMERA"
MODEL_COMPONENT_ID         = (C, A, H, V)
MODEL_COMPONENT_NAME       = ("CENTER", "AXIS",
                              "HORIZONTAL", "VERTICAL")
```

```
MODEL_COMPONENT_UNIT_ID    = ("m", "none", "pixel", "pixel")
MODEL_COMPONENT_1_VECTOR   = (3.469, 14.593, 8.937)
MODEL_COMPONENT_2_VECTOR   = (0.351, 0.758, 17.932)
MODEL_COMPONENT_3_VECTOR   = (14.020, 15.336, 23.714)
MODEL_COMPONENT_4_VECTOR   = (27.423, 3.719, 16.426)
END_OBJECT                 = CAMERA_MODEL
```

In order to facilitate the inclusion of multiple instances of keywords within data product labels without requiring a whole host of new GROUPs, there is a special GROUP called the PARAMETERS GROUP. It has no required elements, and the set of all elements in the PSDD as its optional element set.

```
OBJECT                     = GENERIC_GROUP_DEFINITION
NAME                       = PARAMETERS
STATUS_TYPE                = PENDING
STATUS_NOTE                = "V1.0  2001-07-09 EDR  New Group Definition"
DESCRIPTION                = "The parameters group provides a mechanism for
                             Grouping multiple sets of related parameters
                             within a data product label."

SOURCE_NAME                = "PDS IMG/E. RYE"
REQUIRED_ELEMENT_SET         =                              {}
OPTIONAL_ELEMENT_SET =       {PSDD}

OBJECT                     = ALIAS
NAME                       = "N/A"
USAGE_NOTE                 = "N/A"
END_OBJECT                 = ALIAS

END_OBJECT                 = GENERIC_GROUP_DEFINITION
```

For example:

```
GROUP                      = COMMANDED_INST_PARAMETERS
 SHUTTER_MODE              = "BOTSIM"
 FILTER_NUMBER             = 5
 FILTER_NAME               = "L570-R570"
 EXPOSURE_DURATION         = 1.05
END_OBJECT                 = COMMANDED_INST_PARAMETERS

GROUP                      = TELEMETRY_INST_PARAMETERS
 SHUTTER_MODE              = "AUTO"
 FILTER_NUMBER             = 0
 FILTER_NAME               = "CLEAR"
 EXPOSURE_DURATION         = 0.773
END_OBJECT                 = TELEMETRY_INST_PARAMETERS
```

13.2.1 Implementation of Group Statements

PDS applies the following restrictions to the use of GROUPS:

1. The GROUP structure may only be used in a data product label which also contains one or more data OBJECT definitions.
2. The GROUP statement must contain only attribute assignment statements, include pointers, or related information pointers (i.e., no data location pointers).
3. GROUP statements may not be nested.

4. GROUP statements may not contain OBJECT definitions.
5. Only PSDD elements may appear within a GROUP statement.
6. The keyword contents associated with a specific GROUP identifier (e.g., CAMERA_MODEL) must be identical across all labels of a single data set.

Usage of a GROUP structure must be coordinated with and approved by the responsible PDS discipline Node.

Descriptors may be pre-pended to any generic Group name to produce, and distinguish between, specific instances of the generic group (i.e., any generic Group name may be preceded with a qualifier to uniquely identify the specific instance of the generic Group). For example, the generic PARAMETERS Group could have specific instances of "A_PARAMETERS", "B_PARAMETERS", etc. Pre-pending a descriptor to the generic instances allows multiple instances of the Group to be repeated within a single label.

The specific GROUP is an implementation of the generic GROUP for a particular data set and must be consistent in its structure (i.e., use the same set of keywords) across the data set. For example, the PARAMETERS Group may consist of any keywords defined within the PSDD.

In the following examples, the TELEMETRY_GEOMETRY_PARAMETERS Group consists of three keywords and the CORRECTED_GEOMETRY_PARAMETERS Group consists of three keywords. In this case, both specific instances use the same keywords but could consist of different sets of keywords. Both instances can be collocated within a single data product label. But, each instance across the dataset must contain the same set of keywords.

```
GROUP                       = TELEMETRY_GEOMETRY_PARAMETERS
 GEOMETRY_SOURCE_ID         = "TELEMETRY"
 INSTRUMENT_AZIMUTH         = 35.6 <DEGREES>
 INSTRUMENT_ELEVATION       = -15.4 <DEGREES>
END_OBJECT                  = TELEMETRY_GEOMETRY_PARAMETERS

GROUP                       = CORRECTED_GEOMETRY_PARAMETERS
 GEOMETRY_SOURCE_ID         = "MIPS_MPFMOS"
 INSTRUMENT_AZIMUTH         = 35.9 <DEGREES>
 INSTRUMENT_ELEVATION       = -15.5 <DEGREES>
END_OBJECT                  = CORRECTED_GEOMETRY_PARAMETERS

GROUP                       = CORRECTED_GEOMETRY_PARAMETERS
 GEOMETRY_SOURCE_ID         = "UOFA-BACKLASH"
 INSTRUMENT_AZIMUTH         = 35.8 <DEGREES>
 INSTRUMENT_ELEVATION       = -15.6 <DEGREES>
END_OBJECT                  = CORRECTED_GEOMETRY_PARAMETERS
```

In the near term, the only validation requirements for GROUPs will be that all the elements present in a GROUP must be present in the PDS Data Dictionary. In the future, it is hoped that the contents of the GROUPs will also be validated against their generic GROUP specifications. This would be to ascertain that all the required elements of a particular GROUP are present and that no elements are present that are not specified in the set of required and optional elements.

(This page intentionally left blank.)

Chapter 14. Pointer Usage

Pointers are used within PDS labels to indicate the relative locations of objects in the same file and to reference external files. Pointer statements begin with a caret (“^”) and the name of a PDS object or element. The value part of the pointer statement indicates the location of the referenced information.

14.1 Types of Pointers

Pointer statements fall into three main categories: data location pointers, include pointers, and related information pointers.

14.1.1 Data Location Pointers (Data Object Pointers)

The most common use of pointers is for linking object descriptions to the actual data. The syntax of these pointers depends on whether the label is attached or detached from the data it describes. There are five forms for the value fields, as shown in these examples:

```
(1)    ^IMAGE          = 12
(2)    ^IMAGE          = 600 <BYTES>
(3)    ^INDEX_TABLE    = "INDEX.TAB"
(4)    ^SERIES         = ("C100306.DAT", 2)
(5)    ^SERIES         = ("C100306.DAT", 700 <BYTES>)
```

Examples (1) and (2) are pointers in attached labels. This type of pointer allows reading software to scan the label for the appropriate pointer and then skip right to the data at its location elsewhere in the file. In the first case, the data begin at record 12 of the labeled file. In the second, the data begin at byte 600.

External data files are referenced in examples (3), (4) and (5). Since these pointers occur in detached labels, they must identify a file name and (optional) offset. In example (3), the data begin at record 1 of the data file “INDEX.TAB” (i.e., no explicit offset is taken as an offset of “1”). In example (4), the data begin at record 2 of the data file, "C100306.DAT", whereas in example (5), the data begin at byte 700.

14.1.2 Include Pointers

Another common use of pointers is to reference external files in PDS labels or catalog objects. Files referenced by include pointers are included directly at the location of the pointer statement. These pointers are classified as include-type pointers since they act like the “#include” statements in C program source files. STRUCTURE, CATALOG, and MAP_PROJECTION pointers fall into this category. Following are some examples of include pointer statements:

```
(1)    ^STRUCTURE  = "ENGTAB.FMT"
(2)    ^STRUCTURE  = "IMAGE.FMT"
```

(3) ^CATALOG = "CATALOG.CAT"
(4) ^DATA_SET_MAP_PROJECTION = "DSMAPDIM.CAT"

The structure file in example (1) is referenced by a TABLE object. The “ENGTAB.FMT” file contains column object definitions needed to complete the TABLE definition. Some column definitions might be stored in a separate file if, for example, a number of different TABLE objects use the same definitions. Similarly, in example (2) an IMAGE object definition (i.e., all statements beginning with “OBJECT = IMAGE” and ending with “END_OBJECT = IMAGE”) is contained in an external file called “IMAGE.FMT”.

In example (3), the external file “CATALOG.CAT” is referenced by a VOLUME object in order to provide a full set of catalog information associated with the volume without having to duplicate definitions that already exist in the other file.

In example (4), the external file “DSMAPDIM.CAT” is referenced by an IMAGE_MAP_PROJECTION object to complete the map projection information associated with the image.

14.1.3 Related Information Pointers (Description Pointers)

The third and final use of pointers occurs in PDS labels that reference external files of additional documentation of special use to human readers. These pointers are formed using elements that end in “DESCRIPTION” or “DESC”. They reference text files not written in ODL. Note: These pointers are *not* meant to be used to refer to software tools.

For example:

^DESCRIPTION = "TRK_2_25.ASC"

In this example, the pointer references an external ASCII document file, TRK_2_25.ASC, which provides a detailed description of the data. Note that in this case the documentation file must have its own PDS label, since the label containing the ^DESCRIPTION pointer describes the contents of a different file.

14.2 Rules for Resolving Pointers

Following are the rules for resolving pointer references to external files (see the *Volume Organization and Naming* chapter in this document for information about physical and logical volume structures):

For a pointer statement in FILE_A:

(1) Look in the same directory as FILE_A

(2a) For a single physical volume (no logical volumes), look in the following top level directory:

Pointer	Directory
^STRUCTURE	LABEL
^CATALOG	CATALOG
^DATA_SET_MAP_PROJECTION	CATALOG*
^INDEX_TABLE	INDEX
^DESCRIPTION or ^TEXT	DOCUMENT

(2b) Within a logical volume, look in the top level subdirectory specified by the LOGICAL_VOLUME_PATH_NAME keyword:

Pointer	LOGICAL_VOLUME_PATH_NAME / Directory
^STRUCTURE	LABEL
^CATALOG	CATALOG
^DATA_SET_MAP_PROJECTION	CATALOG*
^INDEX_TABLE	INDEX
^DESCRIPTION or ^TEXT	DOCUMENT

* Note: For volumes using PDS Version 1 or 2 standards, the MAP_PROJECTION files may be located in the LABEL directory

All pointers to data objects should be resolved in step (1), since these files are always required to be located in the same directory as the label file.

(This page intentionally left blank.)

Chapter 15. Record Formats

The choice of proper record format for a data file is influenced by a number of factors. In general, the PDS strongly recommends a record format of fixed-length or stream be used whenever possible to ensure transportability across operating systems and computer platforms and to avoid potential difficulties with interpretation of the underlying data. Records of type FIXED_LENGTH are required for ASCII files described by TABLE Objects. Records of type VARIABLE_LENGTH may be used in cases where storage efficiency is a major consideration, as, for example, in storing compressed images. Records of type STREAM should be used for text files for ease of transportation to various computer systems. Input/output operations with stream files will generally use string-oriented access, retrieving one delimited record from the file each time.

The RECORD_TYPE element in the PDS label indicates the format of the records in the associated data file (attached or detached).

Table 15.1: Recommended Record Formats

	RECORD_TYPE= FIXED_LENGTH	RECORD_TYPE=STREAM	RECORD_TYPE=VARIABLE
Data format	BINARY, ASCII	ASCII	BINARY
Environment	STRUCTURED	AD HOC	STRUCTURED (VAX/VMS)
Data volume	LARGE	SMALL, MEDIUM	VERY LARGE
Input / Output	READ / WRITE	STRING I/O	CUSTOM, SPICE

15.1 FIXED_LENGTH Records

Records of type FIXED_LENGTH normally use a physical record length (RECORD_BYTES) that corresponds directly to the logical record length of the data objects (that is, one physical record for each image line, or one physical record for each row of a table). In some cases, logical records are blocked into larger physical records to provide more efficient storage and access to the data. This blocking is still an important consideration when storing data on magnetic tape, (which requires a gap on the tape between records), but is not generally a consideration in data sets stored on magnetic or CD-ROM disks. In other cases, the physical record length is determined by compatibility with external systems or standards, as in FITS-formatted files.

The PDS strongly recommends using a physical record length that matches the logical record length of the primary data object in the file for greatest compatibility with application software. In the data label, RECORD_BYTES defines the physical record length.

Figure 15.1 illustrates the physical and logical structure used to build a standard PDS FIXED_LENGTH file.

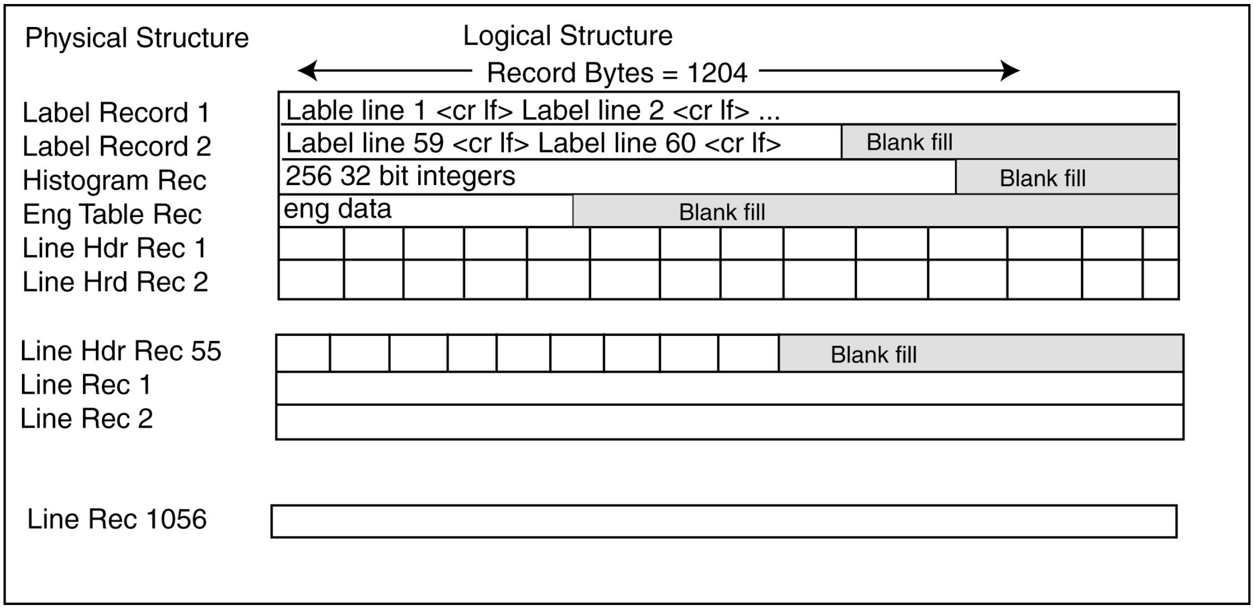


Figure 15.1 Physical and Logical Structure for Fixed Length Files

15.2 STREAM Records

The STREAM record type is reserved for ASCII text files. The records must be delimited by the two-character (carriage return, linefeed) sequence ("<CR><LF>" or "CR/LF"). This is the same record delimiter used for all PDS label and catalog files.

All major operating systems recognize one of either the carriage return, the line feed, or the CR/LF sequence as an ASCII record delimiter; thus, <CR><LF> will work in all cases. There are utilities available for Macintosh (*Apple File Exchange*) and Unix (*tr* translation utility) systems to remove the unneeded extra control character.

Note that the STREAM record type should only be used in those cases where the data contain delimited ASCII records that are *not* of fixed length. The FIXED_LENGTH specification should be used wherever possible.

15.3 VARIABLE_LENGTH Records

PDS data files using the VARIABLE_LENGTH record type must use the VAX/VMS counted byte string format. That is, each record string is preceded by a two-byte LSB integer containing the length of the record. The records may not contain carriage control characters.

The use of the VARIABLE_LENGTH record type is discouraged because of its inherent dependence on *a priori* knowledge of the record structure for proper reading and writing. Notwithstanding, VARIABLE_LENGTH records may be used in the following circumstances:

- When supporting software, which can be executed on a variety of hosts, is provided along with the data. For example, the Voyager CD-ROM disks contain variable-length compressed images along with a decompression program that can be compiled and executed on VAX, PC, Macintosh and UNIX platforms. The decompression program reformats the data into a variety of forms.

- When the files are intended for use only in a specific environment that supports the selected record structure. For example, the Viking Infrared Thermal Mapper (IRTM) CDROM uses a VAX/VMS variable-length record format for software and command files. Note, however, that such proprietary formats are generally inappropriate for PDS deep archiving purposes and should be vigorously avoided in archive volumes.

15.4 UNDEFINED Records

Records with an undefined record type have no specific record structure. For files with attached labels, the label portion should be written using the STREAM conventions described above. When the record type is designated UNDEFINED, no record terminators are recognized and no record length is implied; the data are taken to be a continuous stream of bytes.

The use of the UNDEFINED record type when referring to a single data file is strongly discouraged. "RECORD_TYPE = UNDEFINED" is properly used in cases where a single label points to two or more *different* data files with different record types (i.e., one file with STREAM records and another with VARIABLE_LENGTH records).

(This page intentionally left blank.)

Chapter 16. SFDU Usage

This standard defines restrictions on the use of Standard Formatted Data Units (SFDUs) in archive quality data sets. PDS does not require that data products be packaged as SFDUs. However, if data products are packaged as SFDUs, the following standards apply.

The Consultative Committee for Space Data Systems (CCSDS) has prepared a recommendation for the standardization of the structure and construction rules of SFDUs for the interchange of digital space-related data. An SFDU is a *type-length-value* object. That is, each SFDU consists of: a type identifier which indicates the type of data within the SFDU; a length field which either states the length of the data or indicates how the data are delimited; and a value field which contains the actual data. Both the type and the length fields are included in a 20-byte label, called an *SFDU label* in this document. The value field immediately follows the 20-byte SFDU Label. For PDS data products, this value field is the PDS label, including one or more data object definitions.

There are three versions of SFDUs. In Version 1, the length of an SFDU is represented in binary. In Version 2, the length could also be represented in ASCII. In Version 3, the length can be represented in binary, ASCII, or using one of several delineation techniques. Unless previously negotiated, all PDS data products packaged as SFDUs must be constructed using Version 3 SFDU Labels.

A Version 3 SFDU label consists of the following parts:

l)	Control Authority ID	4 Bytes
2)	Version ID	1 Byte
3)	Class ID	1 Byte
4)	Delimiter Type	1 Byte
5)	Spare	1 Byte
6)	Description Data Unit ID	4 Bytes
7)	Length	8 Bytes

The Control Authority ID and the Description Data Unit ID together form an identifier called an Authority and Description Identifier which points to a semantic (Planetary Science Data Dictionary, in the PDS case) and syntactic (Object Definition Language, 2.0) description of the value field. . The Data Description Unit ID varies by data product type. It is supplied by the JPL Control Authority and is usually documented in the science data product Software Interface Specifications (SIS).

Version 3 allows delimiting of SFDUs either by end-of-file or by start and end markers rather than by explicit byte counts. Further details of the SFDU architecture will not be discussed here. Other sources of information can be found in the *SFDU References* listed in the *Introduction* to this document.

Since archive quality data sets are internally defined, only a limited set of SFDU labels are used to identify the files on a data volume in order to simplify not only the archive products themselves, but also the processing of those products by software. PDS labels are included in the data products, and the information in these PDS labels are considered more than adequate for data identification and scientific analysis.

PDS does not require SFDU labels in its archive products. However, SFDU labels can be accommodated in PDS products when they are required by projects or other agencies concerned in the preparation of the data. The standard use of SFDUs in PDS labels from current missions and data restorations is different from the use of SFDUs in data products from upcoming missions fully supported by the Jet Propulsion Laboratory's Advanced Multi-Mission Operations System (AMMOS). The following sections define the standards for including SFDUs in each case.

Two SFDU organizations are allowed in PDS data products. The first organization (the ZI Structure) has been used historically in PDS data products from restoration and past missions. The second organization (the ZKI organization) is required for data products that pass through the JPL Advanced Multi-Mission Operations System (AMMOS) project database.

16.1 The ZI SFDU Organization

Any PDS data products packaged as SFDUs that are not required to pass through the AMMOS project database as part of an active mission may use the following SFDU organization.

Each instance of a data product (file) in a data set must include two (and only two) SFDU labels. These are a Z Class SFDU label and an I Class SFDU label. The two SFDU labels are concatenated (i.e. Z, then I) and left justified in the first line or record of the PDS label for each data product. (See Figure 16.1.) In the case of data products with detached PDS labels, the two SFDU labels must appear in the first record of the PDS label files and no SFDU labels appear in the data object files. (See Figure 16.2.)

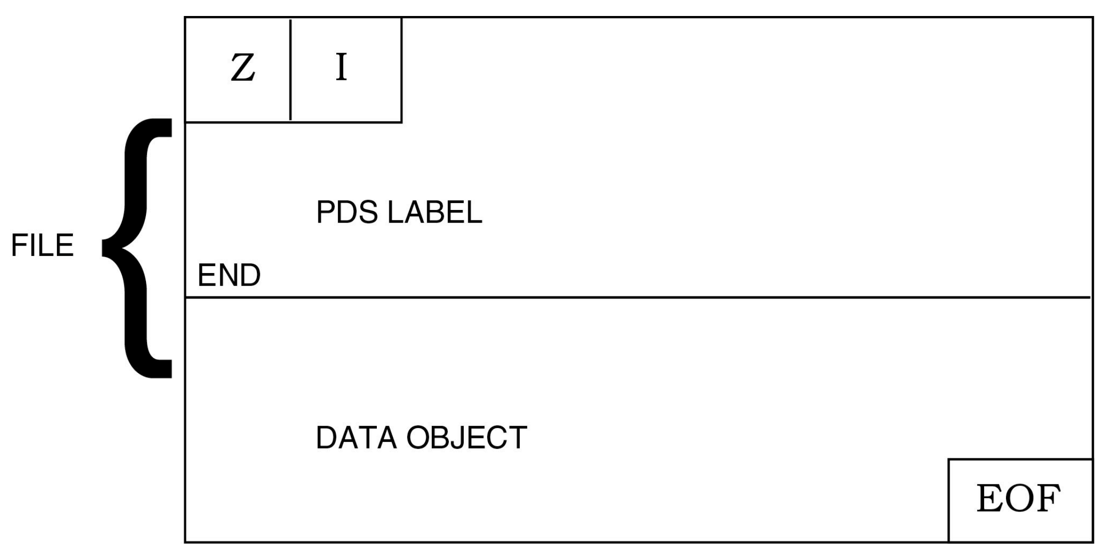


Figure 16.1 Attached PDS Label Example for non-AMMOS compatible products

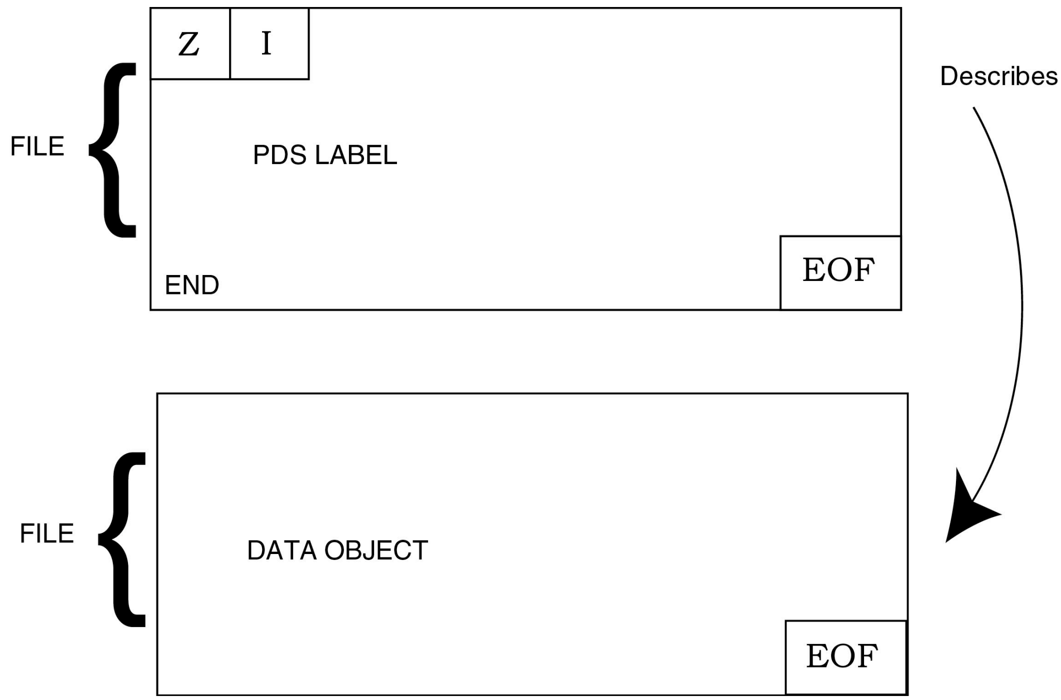


Figure 16.2 Detached PDS Label Example for non-AMMOS compatible products

The first SFDU label must be a Z Class Version 3 SFDU label. “Z Class” indicates that the value field (everything after the first 20 bytes) is an aggregation. In this case, the aggregation consists of only the I Class SFDU. This label also indicates that the delimiter type is End-of-File and that this SFDU (data product) is terminated by a single End-of-File. It is formed as follows:

1)	Control Authority ID	CCSD
2)	Version ID	3
3)	Class ID	Z
4)	Delimiter Type	F
5)	Spare	0
6)	Description Data Unit ID	0001
7)	Length Field	00000001

Example: CCSD3ZF0000l0000000l

The second SFDU label must be an I Class Version 3 SFDU label. “Class I” indicates that the value field (everything after the second 20 bytes) is application data, i.e., the PDS label and the data object(s). The Data Description Unit ID of “PDSX” indicates that the data product uses the

Object Description Language (ODL) syntax and the Planetary Science Data Dictionary semantics to present descriptive information. This SFDU label also indicates that the SFDU (data products) will be terminated by a single End-of-File. It is formed as follows:

1)	Control Authority ID	NJPL
2)	Version ID	3
3)	Class ID	I
4)	Delimiter Type	F
5)	Spare	0
6)	Description Data Unit ID	PDSX
7)	Length Field	00000001

Example: NJPL3IF0PDSX0000000l

```
CCSD3ZF0000100000001NJPL3F0PDSX00000001 <CR> <LF>
PDS_VERSION_ID = PDS3 <CR> <LF>
RECORD_TYPE = STREAM <CR> <LF>
RECORDS = 100 <CR> <LF>
  .
  .
  .
END <CR> <LF>
-------------------------------------------------
DATA OBJECT



                                             EOF
```

Figure 16.3: SFDU Example

The two SFDU labels are concatenated and left justified in the first line or record of the PDS label. Note that there are no characters between the two SFDU labels. See Figure 16.3.

For RECORD_TYPE = STREAM or FIXED_LENGTH or UNDEFINED, the concatenated SFDU labels must be followed immediately by <CR><LF>. For data products that have RECORD_TYPE =VARIABLE_LENGTH, the two SFDU labels may not be followed by <CR><LF>.

STREAM example	CCSD3ZF0000l0000000lNJPL3IF0PDSX0000000l <CR><LF>
FIXED_LENGTH Example	CCSD3ZF0000l0000000lNJPL3IF0PDSX0000000l<CR><LF>
VARIABLE_LENGTH Example	CCSD3ZF0000l0000000lNJPL3IF0PDSX0000000l
UNDEFINED Example	CCSD3ZF0000l0000000lNJPL3IF0PDSX0000000l<CR><LF>

The remainder of the PDS label begins on the next line or record. The last line of the PDS label contains the END statement. Then, if the PDS Label is attached, the data object begins on the next record. If the PDS label is detached, the END statement is the last line of the file.

16.2 The ZKI SFDU Organization

Any PDS data products packaged as SFDUs that are required to pass through the AMMOS project database as part of an active mission must use the following SFDU organization. All data products of this type are assumed to have attached PDS labels.

Each instance of a data product (file) in a data set must include four (and only four) SFDU labels. These are: the Z Class SFDU label; the K Class SFDU label; the End-Marker label for the K Class SFDU; and the I Class SFDU label. The Z and K Class SFDU labels are concatenated (i.e., Z, then K) and left justified in the first line or record of the PDS label for each data product. The End-Marker for the K Class SFDU label and the I Class SFDU label are right justified on the last record of the PDS label (following the END statement). See Figure 16.4.

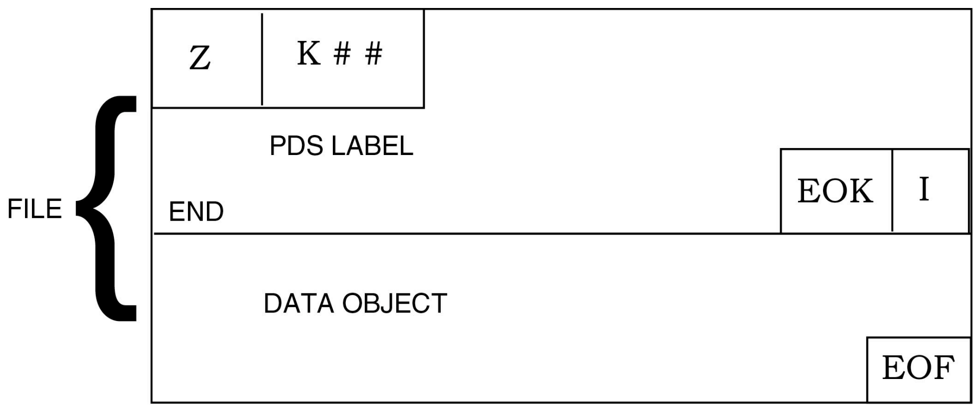


Figure 16.4: PDS Label Example for AMMOS compatible products

The first SFDU label must be a Z Class Version 3 SFDU label. The Z Class indicates that the value field (everything after the first 20 bytes) is an aggregation. In this case, the aggregation consists of a K Class (PDS label) and an I Class (data object) SFDU. This label also indicates that the delimiter type is End-of-File and that this SFDU (data product) is terminated by a single End-of-File. It is formed as follows:

1)	Control Authority	CCSD
2)	Version ID	3
3)	Class ID	Z
4)	Delimiter Type	F
5)	Spare	0
6)	Description Data Unit ID	0001
7)	Length Field	00000001

Example: CCSD3ZF0000l00000001

The second SFDU label must be a K Class Version 3 SFDU label. “Class K” indicates that the value field (everything after the second 20 bytes) is catalog and directory information, i.e., the PDS label (sometimes referred to as the K Header). The Data Description Unit ID of PDSX indicates that the PDS label uses the Object Description Language (ODL) syntax and the Planetary Science Data Dictionary semantics to present data descriptive information. The SFDU label also indicates that the SFDU is delimited by a Start-Marker/End-Marker pair. It is formed as follows:

1)	Control Authority ID	NJPL
2)	Version ID	3
3)	Class ID	K
4)	Delimiter Type	S
5)	Spare	0
6)	Description Data Unit ID	PDSX
7)	Length Field	##mark##

The marker pattern (“##mark##” in the example) can be set to any string that is unlikely to be repeated elsewhere in the data product.

Example: NJPL3KS0PDSX##mark##

The two SFDU labels must be concatenated and left justified in the first line or record of the PDS label. Note that there are no characters between the two SFDU labels. For data products with RECORD_TYPE equal to VARIABLE_LENGTH, the two concatenated SFDU labels must not be followed by <CR><LF>.

Example: CCSD3ZF0000l00000000lNJPL3KS0PDSX##mark##

The remainder of the PDS label begins on the next line. The last line of the PDS label contains the END statement. Then, in the same line or record, right justified, is the End-Marker for the K Class SFDU and the I Class SFDU label. The End-Marker pattern must appear as:

Example: CCSD$$MARKER##mark##

Note that the start marker and the end marker fields must be identical within the SFDU (in the example, “##mark##”). Next must be an I Class Version 3 SFDU label. “Class I” indicates that the value field (everything after the SFDU label) is application data, i.e., the data object. The Data Description Unit ID varies by data product type. It is supplied by the JPL Control Authority and is usually documented in the science data product Software Interface Specifications (SIS). The SFDU label also indicates that the SFDU will be terminated by a single End-of-File. It is formed as follows:

1)	Control Authority ID	NJPL

2)	Version ID	3
3)	Class ID	I
4)	Delimiter Type	F
5)	Spare	0
6)	Description Data Unit ID	XXXX
7)	Length Field	00000001

Example: NJPL3IF001060000000l (where XXXX has been replaced by 0106.)

The two SFDU labels must be concatenated, right justified, and appear in the last line or record of the PDS label following the END statement. (If it happens that there are not 40 bytes left in the last record of the PDS label, add an additional record and right justify the two SFDU labels.) Note that there are no characters between the two SFDU labels, and that the marker pattern and I Class SFDU Labels are transparent to PDS label processing software.

Example: END CCSD$$MARKER##mark##NJPL3IF001060000000l

The data object begins with the next physical record.

16.3 Examples

RECORD_TYPE = STREAM:

End Statement blank(s) End marker I Class SFDU End of record

END CCSD$$MARKER##mark##NJPL3IF0010600000001<CR><LF>

RECORD_TYPE = FIXED_LENGTH:

End Statement Terminator Record Boundary

END <CR><LF> bbbbb CCSD$$MARKER##mark##NJPL3IF0010600000001

RECORD_TYPE = UNDEFINED:

Statement terminator

End Statement
END<CR><LF> CCSD$$MARKER##mark##NJPL3IF0010600000001

RECORD_TYPE = VARIABLE_LENGTH:

Record Length END end of statement

END CCSD$$MARKER##mark##NJPL3IF0010600000001

16.4 Exceptions to this Standard

Software files and document files should not be packaged as SFDUs.

Previous versions of the PDS standards expressed the ZI SFDU labels as an ODL statement. The ZI SFDU labels were followed by "= SFDU_LABEL".

Example: CCSD3ZF0000100000001NJPL3IF0PDSX00000001 = SFDU_LABEL

Chapter 17. Usage of N/A, UNK and NULL

17.1 Interpretation of N/A, UNK, and NULL

During the completion of data product labels or catalog files, one or more values may not be available for some set of required data elements. In this case PDS provides the symbolic literals "N/A", "UNK", and "NULL", each of which is appropriate under different circumstances.

17.1.1 N/A

"N/A" ("Not Applicable") indicates that the values within the domain of this data element are not applicable in this instance. For example, a data set catalog file describing NAIF SPK kernels would contain the line:

```
INSTRUMENT_ID = "N/A"
```

because this data set is not associated with a particular instrument.

"N/A" may be used as needed for data elements of any type (i.e., text, date, numeric, etc.).

17.1.2 UNK

"UNK" ("Unknown") indicates that the value for the data element is not known and never will be. For example, in a data set comprising a series of images, each taken with a different filter, one of the labels might contain the line:

```
FILTER_NAME = "UNK"
```

if the observing log recording the filter name was lost or destroyed and the name of the filter is not otherwise recoverable.

"UNK" may be used as needed for data elements of any type.

17.1.3 NULL

"NULL" is used to flag values that are *temporarily* unknown. It indicates that the data preparer recognizes that a specific value should be applied, but that the true value was not readily available. "NULL" is a placeholder. For example, the line:

```
DATA_SET_RELEASE_DATE = "NULL"
```

might be used in a data set catalog file during the development and review process to indicate that the release date has not yet been determined.

Note that all "NULL" indicators should be replaced by their actual values prior to final archiving of the associated data.

17.2 Implementation Recommendations for N/A, UNK, and NULL

The figurative constants defined above require special values for storage in data base systems. The PDS has the following recommendations for software intended to support PDS labels and catalog objects:

1. In the case of character fields, the explicit string can be stored in the corresponding data elements without further modification. This approach can also be taken where date and time data types are stored as strings.

2. Numeric fields require special flag values to represent the "N/A", "NULL" and "UNK" indicators. Table 17.1 provides suggested standard flag values for each case.

In creating index files based on element values extracted from PDS labels, there are two options for dealing with "N/A", "NULL", and "UNK" in non-string columns:

1. The character strings can be used explicitly in the index. Note, however, that in this case the DATA_TYPE of the column may be forced to "CHARACTER", since, for example, encountering the string "NULL" in what is otherwise a numeric column would cause a read failure.

2. The character strings can be replaced with an appropriate numeric constant. In this case the substitution is indicated in the corresponding column definition by including the NOT_APPLICABLE_CONSTANT, NULL_CONSTANT or UNKNOWN_CONSTANT elements as needed.

Table 17.1: Numeric values for N/A, UNK, NULL

	Signed Integer (4 byte)	**Signed Integer (2 byte)**	**Unsigned Integer (4 byte)**	**Unsigned Integer (2 byte)**	**Tiny Integer (1 byte - unsigned)**	**Real**
N/A	-2147483648	-32768	4294967293	65533	locally defined	-1.E32
UNK	2147483647	32767	4294967294	65534	locally defined	+1.E32
NULL	NULL*	NULL*	NULL*	NULL*	NULL*	NULL*

- "NULL" refers to a system-defined null value. The availability of NULL as a universal value across data types in some data management systems simplifies the implementation of the figurative constant "NULL". However, if a system "null" is not available, then either a) an arbitrary value can be chosen, or b) the meanings of UNK and NULL can be combined and the token or numeric representation of UNK used.

Chapter 18. Units of Measurement

The uniform use of units of measure facilitates broad catalog searches across archive systems.The PDS standard system for units, where applicable, is the *Systeme Internationale d'Unites* (SI). The default units for data elements in the *Planetary Science Data Dictionary* (PSDD) are determined as each element is defined and added to the dictionary. Specific unit definitions are also included in the PSDD.

In cases where more than one type of unit is commonly used for a given data element, an additional data element is provided to explicitly identify the corresponding unit. SAMPLING_PARAMETER_RESOLUTION and SAMPLING_PARAMETER_UNIT are one such pair. The PDS allows exceptions to the SI unit requirement when common usage conflicts with the SI standard (e.g., angles which are measured in degrees rather than radians).

Both singular and plural unit names, as well as unit symbols, are allowed. The double asterisk (**) is used, rather than the caret (^), to indicate exponentiation. When the units associated with a value of a PDS element are not the same as the default units specified in the PSDD (or when explicit units are preferred), a unit expression is used with the value. These unit expressions are enclosed in angular brackets (< >) and follow the value to which they apply.

Examples

```
EXPOSURE_DURATION  = 10 <SECONDS>
DECLINATION        = -14.2756 <DEGREES>
MASS               = 123 <kg>
MASS_DENSITY       = 123 <g/cm**3>
MAP_RESOLUTION     = 123 <PIXEL/DEGREE>
MAP_SCALE          = 123 <KM/PIXEL>
```

Note that in the above example, MASS_DENSITY is not expressed in the SI default unit of measurement for density (kg/m**3).

PDS recommends (in order of preference) that measurements be expressed using the default SI units of measurements, as defined in the following paragraphs. If it is not desirable to use the default SI unit of measurement, then the unit of measurement should be expressed using the SI nomenclature defined in the following paragraphs. If a unit of measurement is not defined by the SI standard, then a unit of measurement can be derived (e.g., pixels per degree, kilometers per pixel, etc.).

18.1 SI Units

The following summary of SI unit information is extracted from *The International System of Units.*

Base units — As the system is currently used, there are seven fundamental SI units, termed “base units”:

QUANTITY	NAME OF UNIT	SYMBOL
length	meter	m
mass	kilogram	kg
time	second	s
electric current	ampere	A
thermodynamic temperature	kelvin	K
amount of substance	mole	mol
luminous intensity	candela	cd

SI units are all written in mixed case; symbols are also mixed case except for those derived from proper names. No periods are used in any of the symbols in the international system.

Derived units — In addition to the base units of the system, a host of derived units, which stem from the base units, are also employed. One class of these is formed by adding a prefix, representing a power of ten, to the base unit. For example, a kilometer is equal to 1,000 meters, and a millisecond is .001 (that is, 1/1,000) second. The prefixes in current use are as follows:

SI PREFIXES

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

Note that the kilogram (rather than the gram) was selected as the base unit for mass for historical reasons. Notwithstanding, the gram is the basis for creating mass units by addition of prefixes.

Another class of derived units consists of powers of base units and of base units in algebraic relationships. Some of the more familiar of these are the following:

QUANTITY	NAME OF UNIT	SYMBOL
area	square meter	m**2
volume	cubic meter	m**3
density	kilogram per cubic meter	kg/m**3
velocity	meter per second	m/s
angular velocity	radian per second	rad/s
acceleration	meter per second squared	m/s**2
angular acceleration	radian per second squared	rad/s**2

kinematic viscosity	square meter per second	m**2/s
dynamic viscosity	newton-second per square meter	N*s/m**2
luminance	candela per square meter	cd/m**2
wave number	1 per meter	m**-1
activity (of a radioactive source)	1 per second	s**-1

Many derived SI units have names of their own:

QUANTITY	NAME OF UNIT	SYMBOL	EQUIVALENT
frequency	hertz	Hz	s**-1
force	newton	N	kg*m/s**2
pressure (mechanical stress)	pascal	Pa	N/m**2
work, energy, quantity of heat	joule	J	N*m
power	watt	W	J/s
quantity of electricity potential difference	coulomb	C	A*s
electromotive force	volt	V	W/A
electrical resistance	ohm	–	V/A
capacitance	farad	F	A*s/V
magnetic flux	weber	Wb	V*s
inductance	henry	H	V*s/A
magnetic flux density	tesla	T	Wb/m**2
luminous flux	lumen	lm	cd*sr
illuminance	lux	lx	lm/m**2

Supplementary units are as follows:

QUANTITY	NAME OF UNIT	SYMBOL
plane angle	radian	rad
solid angle	steradian	sr

Use of figures with SI units — In the international system it is considered preferable to use only numbers between 0.1 and 1,000 in expressing the quantity associated with any SI unit. Thus the quantity 12,000 meters is expressed as "12 km", not "12,000 m". So too, 0.003 cubic centimeters is preferably written "3 mm^3", not "0.003 cm^3".

(This page intentionally left blank.)

Chapter 19. Volume Organization and Naming

The *Volume Organization and Naming* Standard defines the organization of data sets onto physical media and the conventions for forming volume names and identifiers. A *volume* is one unit of a physical medium such as a CD, a DVD, or a magnetic tape. Data sets may reside on one or more volumes and multiple data sets may also be stored on a single volume. Volumes are grouped into *volume sets*.

Each volume has a directory structure containing subdirectories and files. Both random access (CD, DVD) and sequential access (magnetic tape) media are supported. A PDS volume on a sequential access medium has a virtual directory structure defined in the VOLUME object included in the file "VOLDESC.CAT". This virtual structure may then be used to recreate the volume directory structure when the files are moved to a random access medium.

PDS recommends that the entire contents of an archive volume and volume set be based on a single version of the PDS Standards Reference. Software tools that work with one version of the Standards may not work with all versions.

19.1 Volume Set Types

Data may be organized into one of four types of archive volumes, based on the number of data sets on each volume and the number of volumes required to capture all the data. The directory organization of the volumes and the required files varies slightly depending on this volume type. Figures 19.1 through 19.4 depict the various volume directory structure options. The four volume types are described below.

1. *One data set on one volume.* This basic volume organization is illustrated in Figure 19.1. The required and optional files and directories are detailed in Section 19.3.

2. *One data set on many volumes.* In this case the INDEX subdirectory includes both local indices, for the data on the present volume, and cumulative indices, for the data on all (preceding) volumes. This layout is illustrated in Figure 19.2.

3. *Many data sets on one volume.* In this case, additional file naming conventions are imposed to prevent collisions; data subdirectories are organized by data set. There are two variations on this scheme:

    a. *One logical volume* – That is, the data sets collected on the physical medium constitute a single logical volume and would generally be distributed together. See Figures 19.3a and 19.3b, and Section 19.6 for more information on logical volumes.

    b. *Many logical volumes* – The physical medium contains several largely independent collections of data sets, with each collection organized as though it were on its own volume. This is useful when a larger capacity medium (say, DVD) is being used to hold several volumes originally produced on a smaller capacity medium (e.g., CD-ROM). In this case, directories that are common to and identical on all volumes need only be reproduced once (e.g., the SOFTWARE directory in Figure 19.3b). See Figures 19.3a and 19.3b, and Section 19.6 for more information on logical volumes.

4. *Many data sets on many volumes.* This organization is most useful when several large data sets are being produced in parallel over an extended period of time (as with some space missions). Sections of each data set appear on each physical volume, requiring additional naming considerations. See Figure 19.4 for more information.

Note that it is possible to have one or more volumes containing only data accompanied by an ancillary volume containing the DOCUMENT, CATALOG, GAZETTER, SOFTWARE, CALIB, and GEOMETRY directories relevant to all the other volumes. When this is done, the PDS requires that all files referenced by include-type pointers (see the *Pointer Usage* chapter in this document) be present on the data volume. The PDS recommends that ancillary files be archived on the same volume as the corresponding data wherever possible, to facilitate science access.

The contents and organization of the directories of all the volume types are described in the remainder of this chapter.

19.2 Volume Organization Guidelines

The PDS recommends that directory structures be simple, path names short, and directory and file names constructed in a logical manner. When determining the number of files to be stored in each subdirectory, data preparers should keep in mind that most users rely on visual inspection to glean the contents of a directory or confirm that a disk is intact. Note that some older operating systems will “crash” when encountering a directory containing more than 128 files. Note also that device load time can be directly dependent on the number of files in a directory, making large directories inconvenient for large numbers of users. The typical practical limit for these purposes is on the order of 100 files per directory. As a further convenience to users, PDS recommends that empty subdirectories be omitted entirely.

19.3 Description of Directory Contents and Organization

The root directory is the top-level directory of a volume. The following sections describe the contents of the root directory, followed by the contents of the required subdirectories (in alphabetical order), and finally the contents of the optional directories (in alphabetical order).

19.3.1 ROOT Directory Files

AAREADME.TXT **Required**

This file contains an overview of the contents and organization of the associated volume, general instructions for its use, and contact information. The name has been chosen so that it will be listed first in an alphabetical directory listing. See Appendix D for an example of an AAREADME.TXT file.

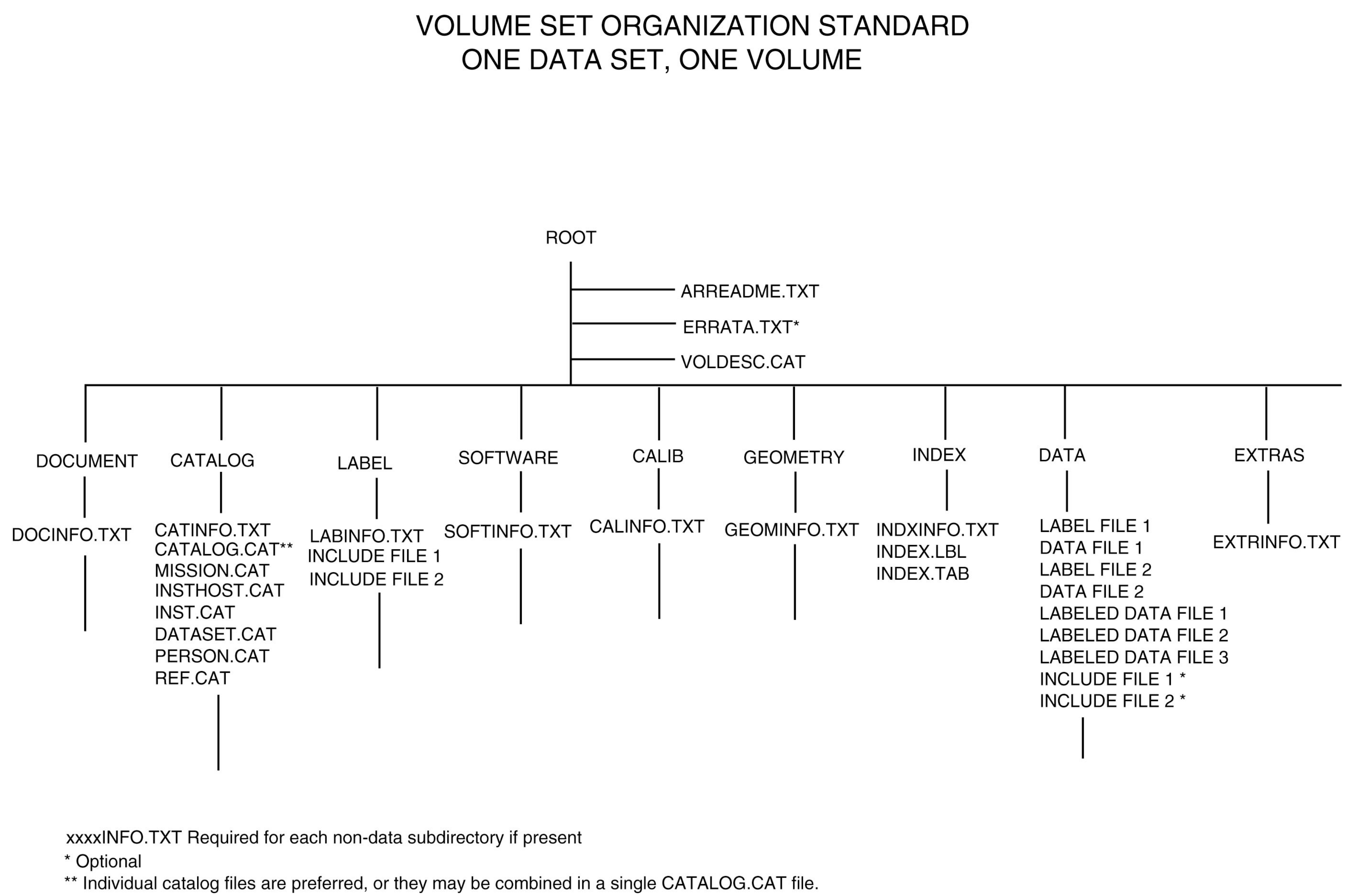


Figure 19.1 Volume Set Organization Standard - One Data Set, One Volume

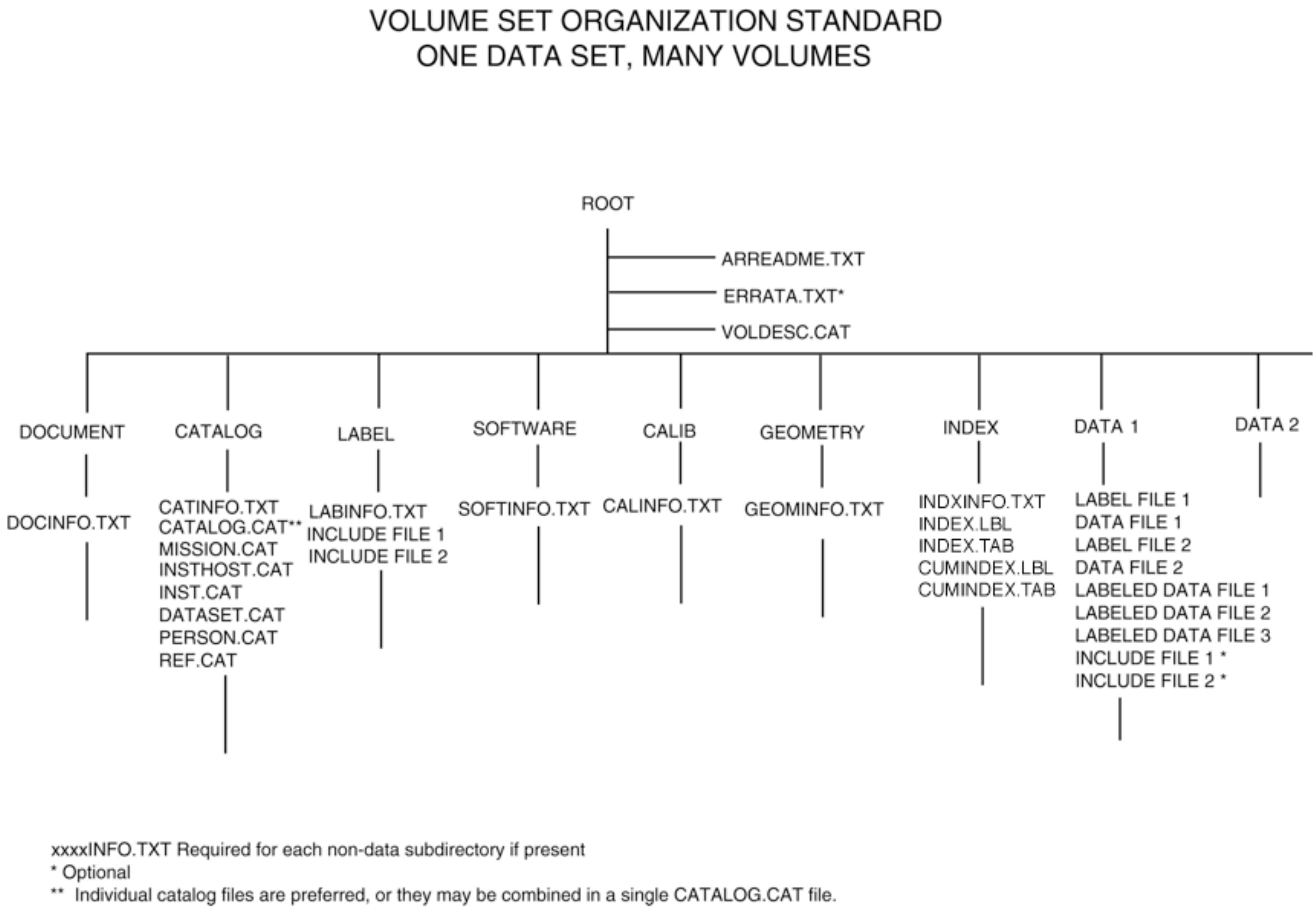
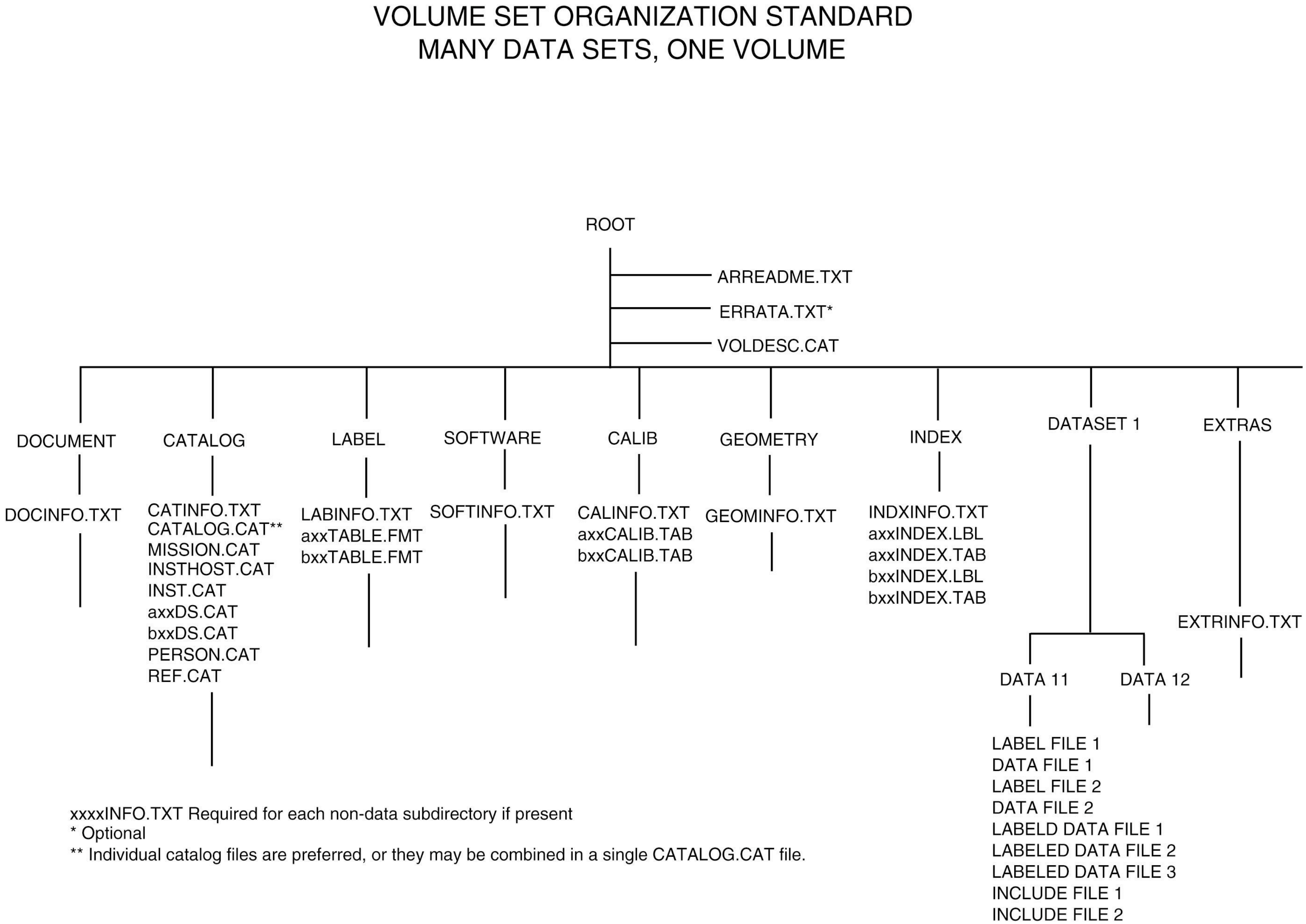


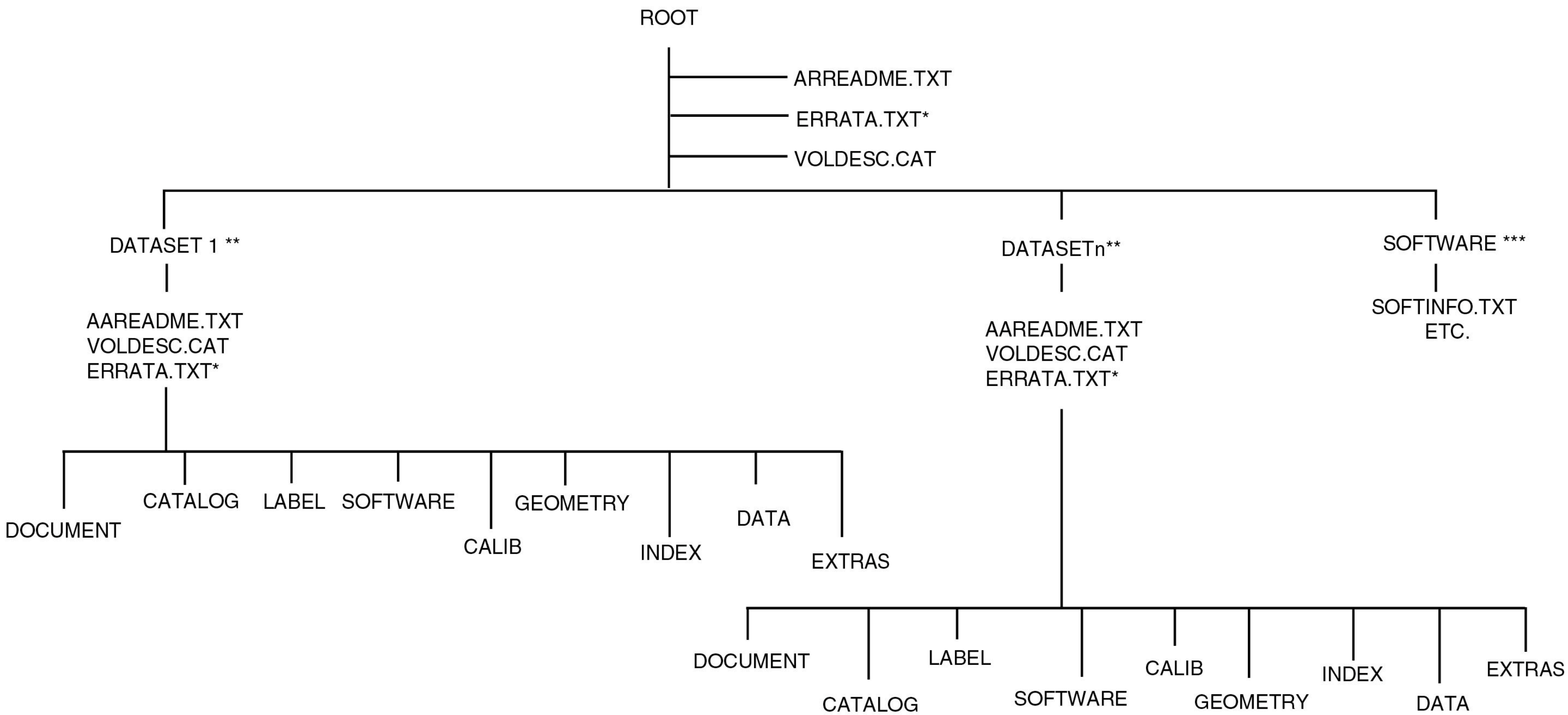
Figure 19.2 Volume Set Organization Standard - One Data Set, Many Volumes



xxxxINFO.TXT Required for each non-data subdirectory if present
* Optional
** Individual catalog files are preferred, or they may be combined in a single CATALOG.CAT file.

Figure 19.3a Volume Set Organization Standard - Many Data Sets, One Volume

VOLUME SET ORGANIZATION STANDARD
MANY DATA SETS, ONE PHYSICAL VOLUME,
MANY LOGICAL VOLUMES



* Optional
** Logical volume; directory structure identical to Figure 19.1, ONE DATA SET, ONE VOLUME
*** Common to all logical volumes

Figure 19.3b Volume Set Organization Standard - Many Data Sets, One Physical Volume, Many Logical Volumes

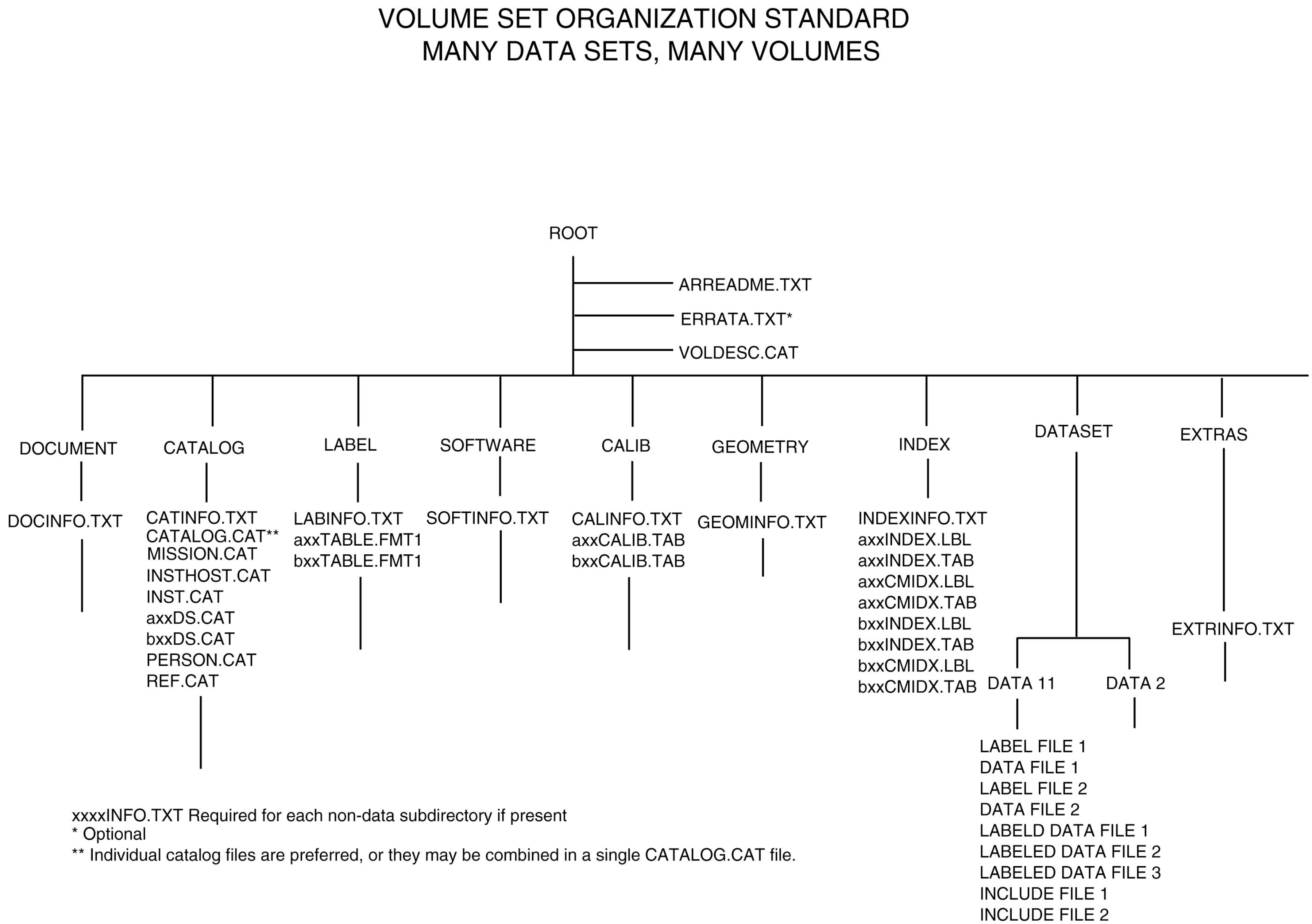


Figure 19.4 Volume Set Organization Standard - Many Data Sets, Many Volumes

VOLDESC.CAT **Required**

This file contains the VOLUME object, which gives a high-level description of the contents of the volume.

ERRATA.TXT **Optional**

This file identifies and describes errors and/or anomalies found in the current volume, and possibly in previous volumes of a set. When a volume contains known errors they must be documented in this file.

VOLDESC.SFD **Obsolete**

This file is identified here only for backward compatibility with previous versions of the PDS standards. It is not to be used in current archive products.

This file contains the SFDU reference object structure that aggregates the separate file contents of the volume into an SFDU. The reference object itself is expressed in ODL. This file should only be included if the data products are packaged as SFDUs. (Note the ".SFD" file extension is a reserved file extension in the CCSDS SFDU standard indicating the file contains a valid SFDU.)

19.3.2 Required Subdirectories

19.3.2.1 CATALOG Subdirectory

This subdirectory contains the catalog object files (for the mission, instrument, data sets, etc.) for the entire volume. When several logical volumes are present on a single physical volume, each logical volume should have its own CATALOG subdirectory.

CATINFO.TXT **Required**

This file identifies and describes the function of each file in the CATALOG subdirectory.

CATALOG.CAT **Optional**

In most cases, the individual catalog objects are in separate files, one for each object. On some older archive volumes, however, all catalog objects were collected into a single file called CATALOG.CAT.

PDS Methodology for Supplying Catalog Objects

The preferred method for supplying catalog objects is as separate files for each catalog object, since this facilitates the review, verification and archiving process. In Figure 19.4, for example, the files *axx*DS.CAT and *bxx*DS.CAT represent two separate files each containing single data set catalog objects (descriptive information about the data set) for data sets *a* and *b* respectively. See the *File Specification and Naming* chapter in this document for the file naming rules; see Section A.5, CATALOG, for the required contents of the catalog object, and see *Appendix B* for information on each of the referenced catalog objects.

When catalog objects are organized in separate files or sets of files, pointer expressions shall be constructed according to the following table. Under "File Name", the first line shows the file name to be used if a single catalog file is present on the volume for the particular type of catalog object named. The second shows the syntax and file name convention to be followed if multiple catalog files are present for the named object.

Catalog Pointer Name		File Name
^DATA_SET_CATALOG	=	"DATASET.CAT"
	=	{"xxxxxxDS.CAT","yyyyyyDS.CAT"}
^DATA_SET_COLLECTION_CATALOG	=	"DSCOLL.CAT"
	=	{"xxxxxDSC.CAT","yyyyyDSC.CAT"}
^DATA_SET_MAP_PROJECTION_CATALOG	=	"DSMAP.CAT"
	=	{"xxxDSMAP.CAT","yyyDSMAP.CAT"}
^INSTRUMENT_CATALOG	=	"INST.CAT"
	=	{"xxxxINST.CAT","yyyyINST.CAT"}
^INSTRUMENT_HOST_CATALOG	=	"INSTHOST.CAT"
	=	{"xxxxHOST.CAT","yyyyHOST.CAT"}
^MISSION_CATALOG	=	"MISSION.CAT"
	=	{"xxxxxMSN.CAT","yyyyyMSN.CAT"}
^PERSONNEL_CATALOG	=	"PERSON.CAT"
	=	{"xxxxPERS.CAT,"yyyyPERS.CAT"}
^REFERENCE_CATALOG	=	"REF.CAT"
	=	{"xxxxxREF.CAT","yyyyyREF.CAT"}
^SOFTWARE_CATALOG	=	"SOFTWARE.CAT"
	=	{"xxxSW.CAT", "yyySW.CAT"}
^TARGET_CATALOG	=	"TARGET.CAT"
	=	{"xxxTGT.CAT", "yyyTGT.CAT"}

19.3.2.2 Data Subdirectory

The DATA subdirectory may be used to unclutter the root directory of a volume by providing a single entry point to multiple data subdirectories. These directories contain the data product files. The directories are organized and named according to the standards in Chapter 8, *Directory Types and Naming*, in this document. Subdirectories may be nested up to eight levels deep on a physical volume.

Data Files

A data file contains one or more data objects, which is a grouping of data resulting from a scientific observation (such as an image or table) and representing the measured instrument parameters.

Label Files

A label file contains a detached PDS label that identifies, describes, and defines the structure of the data objects. The associated data objects are contained in an accompanying data file. The label file must have the same base name as the associated data file, with an extension of “.LBL”.

Labeled Data Files

PDS labels may be attached directly to the data they describe. In this case the PDS label comes first and the data begin immediately following the end of the label. When attached labels are used, no ".LBL" files will be present in the data directories. See the *Data Products* and *Data Product Labels* chapters in this manual for details.

19.3.2.3 INDEX Subdirectory

This directory contains the indices for all data products on the volume.

Note: If the physical volume is organized as several logical volumes (case *3b* of Section 19-1), there will generally not be an INDEX subdirectory at the root of the physical volume. Instead there will be individual INDEX subdirectories at the root of each logical volume. See Section A.21, INDEX_TABLE, for more information.

INDXINFO.TXT **Required**

This file identifies and describes the function of each file in the INDEX subdirectory. This description should include at least:

1) A description of the structure and contents of each index table in this subdirectory

2) Usage notes

For an example of the INDXINFO.TXT file, see Appendix D, Section D.2.

INDEX.LBL **Required**

This is the PDS label for the volume index file, INDEX.TAB. The INDEX_TABLE specific object should be used to identify and describe the columns of the index table. See *Appendix A* for an example. Although INDEX.LBL is the preferred name for this file, the name *axx*INDEX.LBL may also be used (with *axx* replaced by an appropriate mnemonic).

Note: The PDS recommends detached labels for index tables. If an attached label is used, this file is omitted.

INDEX.TAB **Required**

This file contains the volume index in tabular format (i.e., the INDEX_TABLE specific object is used to identify and describe the data stored on an archive volume). Only data product label files (i.e., not the data files) are included in an index table. In rare cases, however, ancillary files are also included. Although INDEX.TAB is the preferred name for this file, the name *axx*INDEX.TAB may also be used, with *axx* replaced by an appropriate mnemonic.

Note that the *axx* prefix is neither required nor recommended. Data producers may use a prefix to distinguish two or more files by data set, instrument, or other criteria. The data producer should replace the generic prefixes shown here with a suitable mnemonic.

The following files are recommended for multi-volume sets:

CUMINDEX.LBL **Optional**

This file contains the cumulative volume set index in tabular format (i.e., the INDEX_TABLE specific object is used to identify and describe the data stored on each archive volume). Only data product label files (i.e., not the data files) are included in an index table. In rare cases, however, ancillary files may be included. Although CUMINDEX.LBL is the preferred name for this file, the name *axx*CMIDX.LBL may also be used, with *axx* replaced by an appropriate mnemonic.

PDS recommends the use of detached labels for index tables. If an attached label is used, this file is omitted.

CUMINDEX.TAB **Optional**

This file contains the cumulative volume set index in a tabular format. Normally only data files are included in a cumulative index table. In some cases, however, ancillary files may be included. Although CUMINDEX.TAB is the preferred name for this file, the name *axx*CMIDX.TAB may also be used, with *axx* replaced by an appropriate mnemonic.

19.3.3 Optional Subdirectories

19.3.3.1 CALIBration Subdirectory

This directory contains the calibration files used in the processing of the raw data or needed to use the data products on the volume. Note that “CALIB” is only a recommended name - a different directory name may be used if appropriate.

CALINFO.TXT **Required**

This file identifies and describes the function of each file in the CALIB subdirectory.

Calibration Files **Required**

In Figures 19.3a and 19.4, the files *axx*CALIB.TAB and *bxx*CALIB.TAB represent sample files. The *axx* and *bxx* prefixes indicate that the calibration files for different data sets (*a* and *b*) may be combined in the same CALIB subdirectory.

Note that the *axx* and *bxx* prefixes in the sample names are neither required nor recommended. Data producers may use them to distinguish two or more files (by data set, instrument, or other criteria). Also, in this case the “CALIB” file name is not required. It is used in the figures to differentiate calibration files from observational data files. The data producer should replace the generic file names shown here by suitably mnemonic names.

19.3.3.2 DOCUMENT Subdirectory

This directory contains the files that provide documentation and supplementary and ancillary information to assist in understanding and using the data products on the volume. The documentation may describe the mission, spacecraft, instrument, and data set(s). It may include references to science papers published elsewhere as well an entire papers republished on the volume. See Section A.12, DOCUMENT, for more information.

DOCINFO.TXT **Required**

This file identifies and describes the function of each file in the DOCUMENT subdirectory.

VOLINFO.TXT **Optional**

This file describes the attributes and contents of the volume. This file is sometimes included in addition to the catalog files in the CATALOG subdirectory to provide the same information in an alternate format.

Note: In rare cases, the data engineer may allow the data preparer to place all the corresponding catalog object descriptions in the VOLINFO.TXT file of the DOCUMENT subdirectory in lieu of separate files in the CATALOG subdirectory. Regardless of which method is used, the descriptions themselves must always be supplied.

Data Dictionary Files **Optional**

The data dictionary files are comprised of two files, PDSDD.FUL and PDSDD.IDX. The PDSDD.FUL file identifies and describes the data object and data element definitions contained in the Planetary Science Data Dictionary (PSDD). The PDSDD.IDX is an index of the PDSDD.FUL and is currently used by the PDS validation tools to quickly locate individual elements in the PSDD.

These files are human-readable ASCII text and are useful for (future) users to ascertain the data object and data element definitions used within the PDS at the time that the archive product was produced.

The above files are required if locally-defined data elements are used in the archive product, and are recommended if the archive product does not use locally-defined data elements.

The PDSDD.FUL and PDSDD.IDX files can be labeled using either the TEXT or ASCII_DOCUMENT objects.

Example: PDSDD.LBL

```
PDS_VERSION_ID    = PDS3
RECORD_TYPE       = STREAM

^FUL_TEXT         = "PDSDD.FUL"
```

```
^IDX_TEXT         = "PDSDD.IDX"

OBJECT            = FUL_TEXT
  PUBLICATION_DATE = 2003-12-31
END_OBJECT        = FUL_TEXT

OBJECT            = IDX_TEXT
  PUBLICATION_DATE = 2003-12-31
END_OBJECT        = IDX_TEXT
END
```

19.3.3.3 EXTRAS Subdirectory

The EXTRAS directory is the designated area for housing additional elements provided by data preparers beyond the scope of the PDS archive requirements. Examples include HTML-based disk navigators, educational and public interest aids, and other useful but nonessential items. The PDS places no restrictions on the contents and organization of this subdirectory other than conformance to ISO-9660/UDF standards.

EXTRINFO.TXT **Required**

This file identifies and describes the function of each file in the EXTRAS subdirectory. This description should include at least the following:

1. A description of the structure and contents of each file in the subdirectory
2. Usage notes

19.3.3.4 GAZETTER Subdirectory

This directory contains detailed information about all the named features on a target body (i.e., the gazetteer information) associated with the data sets on the volumes. “Named features” are those the International Astronomical Union (IAU) has named and approved. See Section A.16, GAZETTER_TABLE, for more information.

GAZINFO.TXT **Required**

This file identifies and describes the function of each file in the GAZETTER subdirectory.

GAZETTER.TXT **Required**

This file contains text describing the structure and contents of the gazetteer table in GAZETTER.TAB.

GAZETTER.LBL **Required**

This file is the PDS label containing a formal description of the structure of the gazetteer table.

GAZETTER.TAB **Required**

This file contains the gazetteer table.

19.3.3.5 GEOMETRY Subdirectory

This directory contains the files (e.g., SEDR file, SPICE kernels, etc.) needed to describe the observation geometry for the data. Note that "GEOMETRY" is only a recommended directory name, another appropriate name may be used.

GEOMINFO.TXT **Required**

This file identifies and describes the function of each file in the GEOMETRY subdirectory.

19.3.3.6 LABEL Subdirectory

This directory contains additional PDS labels and include files that were not packaged with the data products or in the data subdirectories. When multiple logical volumes reside on a single physical volume, the LABEL subdirectories *must* appear below the logical volume root directories. This is because the rules governing pointer resolution preclude a search across logical volumes.

LABINFO.TXT **Required**

This file identifies and describes the function of each file in the LABEL subdirectory.

Include Files **Required**

Include files are files referenced by a pointer in a PDS label. Typically they contain additional metadata or descriptive information. Only files of type LBL, TXT, or FMT ("format") may be included in the LABEL subdirectory. In Figures 19.1-4, the files *axx*INCLUDE FILE1, *bxx*INCLUDE FILE1 and INCLUDE FILE1 represent sample files of the above types. The *axx* and *bxx* prefixes indicate that the include files for different data sets (*a* and *b*) may be combined in the same LABEL subdirectory.

Note that the *axx* and *bxx* prefixes in the sample names are neither required nor recommended. Data producers may use them to distinguish two or more files (by data set, instrument, or other criteria). The data producer should replace the generic prefixes shown here by a suitable mnemonic.

19.3.3.7 SOFTWARE Subdirectory

This directory contains the software libraries, utilities, or application programs supplied for accessing or processing the data. It may also include descriptions of processing algorithms. Only public domain software may be included on PDS archive volumes.

Two subdirectory structures are available for organizing the SOFTWARE directory: platform-based and application-based. Platform-based is the recommended method for general archives and is described below. For an example of application-based organization see the example for

SOFTINFO.TXT in *Appendix D* of this document, and the NAIF directory structure in *Appendix E*. See Section 11.3 for information about packaging software for inclusion in an archive product.

SOFTINFO.TXT **Required**

This file identifies and describes the function of each file in the SOFTWARE subdirectory.

SRC Subdirectory **Optional**

There can be a global SRC directory under the SOFTWARE directory if there is source code applicable to all platforms. For example, application-programming languages such as IDL are relatively platform independent and would be placed in a global SRC directory. Note that in the example below, there is both a global source directory as well as source directories at the lower levels.

DOC Subdirectory **Optional**

This directory contains documentation for the software in the parallel SRC directory.

LIB Subdirectory **Optional**

This directory contains libraries applicable to all platforms.

Hardware Platform and Operating System/Environment Subdirectories **Optional**

If only global source code is being provided on the volume, no further organization is required. If platform- or environment- specific software is being provided, the structure in Figure 19.5 should be followed. Specifically:

1. The hardware platform and the operating system/environment must be explicitly stated. If more than one operating system/environment (OS/Env) is supported for a single hardware platform, each should have its own subdirectory under the hardware directory. If there is only one, then that subdirectory can be promoted to the hardware directory level (via naming conventions). In Figure 19.5, several environments are supported for platform HW1, but only one for HW2 – thus the difference in subdirectory structures.
2. The next directory level contains BIN, SRC, DOC, LIB and OBJ. If any of these are not applicable, it should be left out (i.e., empty directories should be omitted).
3. Following are examples of subdirectory names for both multiple and single OS/Env per platform. (This list is provided for illustration only. It is not meant to be exhaustive.)

Multiple	Single
PC	
DOS	PCDOS
WIN	PCWIN
WINNT	PCWINNT
OS2	PCOS2

MAC	
SYS7	MACSYS7
AUX	MACAUX
SUN	
SUNOS	SUNOS
SOLAR	SUNSOLAR
VAX	
VMS	VAXVMS
ULTRX	VAXULTRX
SGI	
IRX4	SGIIRX4
IRX5	SGIIRX5

19.4 Volume Naming

Volume names must be no more than 60 characters in length and in upper case. They should describe the contents of the volume in terms that a human user can understand. In most cases the volume name is more specific than the volume set name. For example, the volume name for the first volume in the *VOYAGER IMAGES OF URANUS* volume set is "VOLUME 1: COMPRESSED IMAGES 24476.54 - 26439.58."

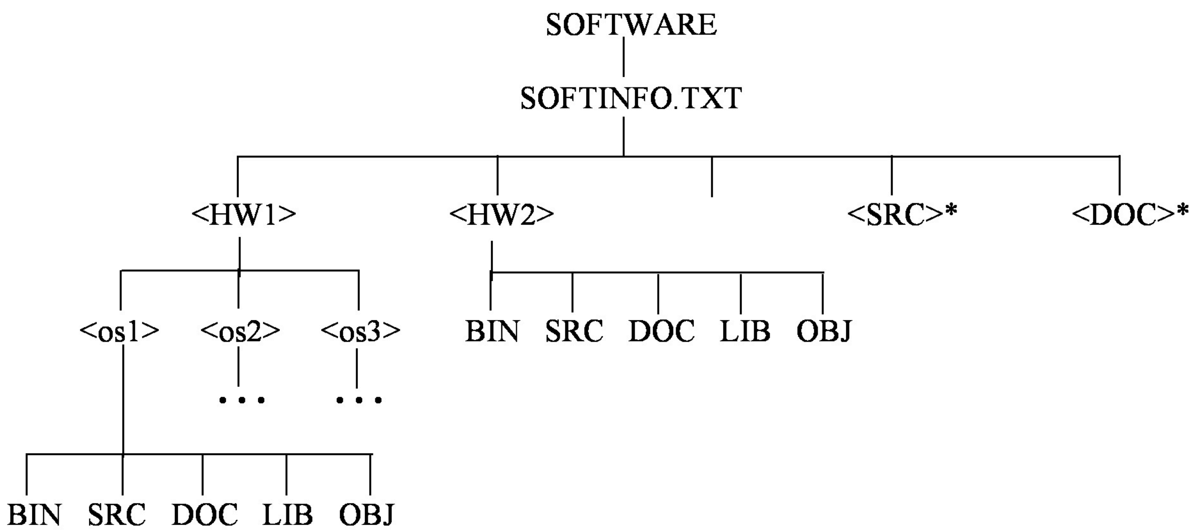
19.4.1 Volume ID

Many types of media and the machines that read them place a limit on the length of the volume ID. Therefore, although the complete volume set ID should be placed on the outside label of the volume, a shorter version is actually used when the volume is recorded. PDS has adopted a limit of 11 characters for these terse volume identifiers. This volume ID consists of the last two components of the volume set ID, with the "X" wildcard values replaced by the sequence number associated with the particular volume (see the *Volume Set ID* Standard below). This ID must always be unique for PDS data volumes. The volume ID must be in upper case.

Examples:

VG_0002	Volume 2 of the Voyager set
MG_0001	The first volume of the Magellan set
VGRS_0001	A potential Voyager Radio Science collection

If a volume is redone because of errors in the initial production the volume ID should remain the same and the VOLUME_VERSION_ID incremented. This parameter is contained in the VOLDESC.CAT file on the volume. The version ID should also be placed on the external volume label as "Version *n*" where *n* indicates the revision number. A revision number greater than one indicates that the original volume should be replaced with the new version.



* **NOTE:** INFO.TXT files under SOFTWARE subdirectories are optional (e.g., PCINFO.TXT, MACINFO.TXT, VAXINFO.TXT, SUNINFO.TXT, etc.).

Figure 19.5 – Platform-based SOFTWARE Subdirectory Structure

19.5 Volume Set Naming

The volume set name provides the full, formal name of a group of data volumes containing one or a collection of related data sets. Volume set names may be at most 60 characters in length and must be in upper case. Volume sets are normally considered a single orderable entity. For example, the volume series MISSION TO VENUS consists of the following volume sets:

MAGELLAN: THE MOSAIC IMAGE DATA RECORD

MAGELLAN: THE ALTIMETRY AND RADIOMETRY DATA RECORD

MAGELLAN: THE GLOBAL ALTIMETRY AND RADIOMETRY DATA RECORD

PRE-MAGELLAN RADAR AND GRAVITY DATA SET COLLECTION

In certain cases, the volume set name can be the same as the volume name, e.g., when the volume set consists of only one volume.

19.5.1 Volume Set ID

A *volume set* is a series of archive volumes that are closely related. In general, the volumes of a set will be distributed and used together. Each volume within the set must have a VOLUME_ID that is unique across the PDS archive. The volume set is identified by a VOLUME_SET_ID of up to 60 characters incorporating the range of constituent VOLUME_IDs. VOLUME_SET_IDs must be in upper case, and are composed by concatenating the following fields, separated by underscores, using abbreviations if necessary:

1. The country of origin (abbreviated)
2. The government branch
3. The discipline within the branch that is producing the volumes
4. A campaign, mission or spacecraft identifier followed by an optional instrument or product identifier (6 characters)
5. A 4-digit sequence identifier: The first digit(s) represent the volume set; the remaining digits contain “X”, representing the range of volumes in the set. Up to four “X” characters may be used.

Example

USA_NASA_PDS_GO_10XX could be the volume set ID for the Galileo EDR volume set, since there are less than 100 volumes (since the XX placeholder accommodates the range 01 - 99 only). Volume IDs for volumes in the set would then be GO_1001, GO_1002, etc.

Note: Because of the uniqueness constraint, data preparers should consult with their PDS data engineer when it comes time to formulate new VOLUME_ID and VOLUME_SET_ID values.

Volume Set IDs Prior to PDS Version 3.2

Prior to version 3.2, the 4-digit sequence identifier (item 5 above) did not include the “X” wildcards. Instead, the last digits represented the volume. For example, on Magellan, a volume set ID “USA_NASA_JPL_MG_0001” was used *only* for the volume with the volume ID “MG_0001”. Subsequent volumes in the same set had volume set IDs that differed in the final field. When a set of volumes was to be distributed as one logical unit, the volume set ID included the range of volume IDs.

Example

USA_NASA_PDS_VG_0001_TO_VG_0003 for the three volumes that comprise the Voyager Uranus volume set.

19.6 Logical Volume Naming

Logical volumes retain the volume and volume set naming used at the physical volume level. For further information, see the “Volume Object” in *Appendix A* of this document.

19.7 Exceptions to This Standard

In rare cases volume IDs are subject to restrictions imposed by specific hardware or software environments. Also, volumes made in the past may have IDs that do not meet this standard and there may be compelling reasons for keeping the same volume ID when making a new copy of the data. All new data sets, however, must adhere to this standard wherever possible.

Appendix A. PDS Data Object Definitions

This section provides an alphabetical reference of approved PDS data object definitions used for labeling primary and secondary data objects. The definitions include descriptions, lists of required and optional keywords, lists of required and optional subobjects (or child objects), and one or more examples of specific objects. For a more detailed discussion on primary and secondary data objects, see the *Data Products* chapter in this document.

Data object definitions are refined and augmented from time to time, as user community needs arise, so object definitions for products designed under older versions of the Standards may differ significantly. To check the current state of any object definition, consult a PDS data engineer or either of these URLs:

PDS Catalog Search: **http://pdsproto.jpl.nasa.gov/onlinecatalog/top.cfm**

Data Dictionary Search:
http://pds.jpl.nasa.gov/tools/ddlookup/data_dictionary_lookup.cfm

The examples provided in this Appendix are based on both existing and planned PDS archive products, modified to reflect the current version of the PDS Standards. Additional examples may be obtained by contacting a PDS Data Engineer.

NOTE: Any keywords in the *Planetary Science Data Dictionary* may also be included in a specific data object definition.

Primitive Objects

There exist four primitive data objects: ARRAY; BIT_ELEMENT; COLLECTION; and ELEMENT. Although these objects are available, they should only be used after careful consideration of the current high-level PDS Data Objects. Please see the *PDS Objects* chapter in this document for guidelines on the use of primitive objects.

Chapter Contents

Appendix A. PDS Data Object Definitions A-1

A.1 ALIAS A-3
A.2 ARRAY (Primitive Data Object) A-4
A.3 BIT_COLUMN A-8
A.4 BIT ELEMENT (Primitive Data Object) A-11
A.5 CATALOG A-12
A.6 COLLECTION (Primitive Data Object) A-15
A.7 COLUMN A-16
A.8 CONTAINER A-20
A.9 DATA_PRODUCER A-27
A.10 DATA_SUPPLIER A-29
A.11 DIRECTORY A-31
A.12 DOCUMENT A-33
A.13 ELEMENT (Primitive Data Object) A-36
A.14 FIELD A-38
A.15 FILE A-41
A.16 GAZETTEER_TABLE A-45
A.17 HEADER A-55
A.18 HISTOGRAM A-57
A.19 HISTORY A-60
A.20 IMAGE A-64
A.21 INDEX_TABLE A-69
A.22 PALETTE A-74
A.23 QUBE A-77
A.24 SERIES A-85
A.25 SPECTRAL_QUBE A-90
A.26 SPECTRUM A-107
A.27 SPICE KERNEL A-110
A.28 SPREADSHEET A-113
A.29 TABLE A-118
A.30 TEXT A-139
A.31 VOLUME A-141
A.32 WINDOW A-148

A.1 ALIAS

The ALIAS object provides a method for identifying alternate terms or names for approved data elements or objects within a data system. The ALIAS object is an optional sub-object of the COLUMN object.

A.1.1 Required Keywords

1. ALIAS_NAME
2. USAGE_NOTE

A.1.2 Optional Keywords

Any

A.1.3 Required Objects

None

A.1.4 Optional Objects

None

A.1.5 Example

The following label fragment shows the ALIAS object included as a sub-object of a COLUMN:

```
OBJECT                        = COLUMN
  NAME                        = ALT_FOOTPRINT_LONGITUDE
  START_BYTE                  = 1
  DATA_TYPE                   = REAL
  BYTES                       = 10

  OBJECT                      = ALIAS
    ALIAS_NAME                = AR_LON
    USAGE_NOTE                = "MAGELLAN MIT ARCDR SIS"
  END_OBJECT                  = ALIAS
END_OBJECT                    = COLUMN
```

A.2 ARRAY (Primitive Data Object)

The ARRAY object is provided to describe dimensioned arrays of homogeneous objects. Note that an ARRAY may contain only a single sub-object, which can itself be another ARRAY or COLLECTION if required. A maximum of 6 axes is allowed in an ARRAY. By default, the rightmost axis is the fastest varying axis.

The optional "AXIS_*" elements are used to describe the variation between successive objects in the ARRAY. Values for AXIS_ITEMS and "AXIS_*" elements for multidimensional arrays are listed in axis order. The optional START_BYTE data element provides the starting location relative to an enclosing object. If a START_BYTE is not specified, a value of 1 is assumed.

A.2.1 Required Keywords

1. AXES
2. AXIS_ITEMS
3. NAME

A.2.2 Optional Keywords

1. AXIS_INTERVAL
2. AXIS_NAME
3. AXIS_UNIT
4. AXIS_START
5. AXIS_STOP
6. AXIS_ORDER_TYPE
7. CHECKSUM
8. DESCRIPTION
9. INTERCHANGE_FORMAT
10. START_BYTE

A.2.3 Required Objects

None

Note that while no specific sub-object is required, the ARRAY object must contain at least one of the optional objects, following. That is, a null ARRAY object may not be defined.

A.2.4 Optional Objects

1. ARRAY
2. BIT_ELEMENT
3. COLLECTION
4. ELEMENT

A.2.5 Example 1

Following is an example of a two-dimensional spectrum array in a detached label.

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 1600
FILE_RECORDS                  = 180

DATA_SET_ID                   = "IHW-C-SPEC-2-EDR-HALLEY-V1.0"
OBSERVATION_ID                = "704283"
TARGET_NAME                   = "HALLEY"
INSTRUMENT_HOST_NAME          = "IHW SPECTROSCOPY AND SPECTROPHOTOMETRY
                                 NETWORK"
INSTRUMENT_NAME               = "IHW SPECTROSCOPY AND
                                 SPECTROPHOTOMETRY"
PRODUCT_ID                    = "704283"
OBSERVATION_TIME              = 1986-05-09T04:10:20.640
START_TIME                    = 1986-05-09T04:07:50.640
STOP_TIME                     = UNK
PRODUCT_CREATION_TIME         = 1993-01-01T00:00:00.000
^ARRAY                        = "SPEC2702.DAT"

/* Description of Object in File */

OBJECT                        = ARRAY
  NAME                        = "2D SPECTRUM"
  INTERCHANGE_FORMAT          = BINARY
  AXES                        = 2
  AXIS_ITEMS                  = (180,800)
  AXIS_NAME                   = ("RHO","APPROXIMATE WAVELENGTH")
  AXIS_UNIT                   = (ARCSEC,ANGSTROMS)
  AXIS_INTERVAL               = (1.5,7.2164)
  AXIS_START                  = (1.0,5034.9)

  OBJECT                      = ELEMENT
    DATA_TYPE                 = MSB_INTEGER
    BYTES                     = 2
    NAME                      = COUNT
    DERIVED_MAXIMUM           = 2.424980E+04
    DERIVED_MINIMUM           = 0.000000E+00
    OFFSET                    = 0.000000E+00
    SCALING_FACTOR            = 1.000000E+00
```

```
      NOTE                          = "Conversion factor 1.45 may be applied
                                       to data to estimate photons/sq
                                       m/sec/angstrom at 6800 angstroms."
      END_OBJECT                    = ELEMENT
    END_OBJECT                      = ARRAY
  END
```

A.2.6 Example 2

The following label shows ARRAY, COLLECTION and ELEMENT primitive objects all used together.

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 122
FILE_RECORDS                  = 7387

^ARRAY                        = "MISCHA01.DAT"

DATA_SET_ID                   = "VEGA1-C-MISCHA-3-RDR-HALLEY-V1.0"
TARGET_NAME                   = HALLEY
SPACECRAFT_NAME               = "VEGA 1"
INSTRUMENT_NAME               = "MAGNETOMETER"
PRODUCT_ID                    = "XYZ"
START_TIME                    = "UNK"
STOP_TIME                     = "UNK"
SPACECRAFT_CLOCK_START_COUNT  = "UNK"
SPACECRAFT_CLOCK_STOP_COUNT   = "UNK"

NOTE                          = "VEGA 1 MISCHA DATA"

OBJECT                        = ARRAY
  NAME                        = MISCHA_DATA_FILE
  INTERCHANGE_FORMAT          = BINARY
  AXES                        = 1
  AXIS_ITEMS                  = 7387
  DESCRIPTION                 = "This file contains an array of fixed-
                                 length Mischa records."

  OBJECT                      = COLLECTION
    NAME                      = MISCHA_RECORD
    BYTES                     = 122
    DESCRIPTION               = "Each record in this file consists of a
                                 time tag followed by a 20-element array
                                 of magnetic field vectors."

    OBJECT                    = ELEMENT
      NAME                    = START_TIME
      BYTES                   = 2
      DATA_TYPE               = MSB_INTEGER
      START_BYTE              = 1
    END_OBJECT                = ELEMENT
```

```
    OBJECT                    = ARRAY
      NAME                    = MAGNETIC_FIELD_ARRAY
      AXES                    = 2
      AXIS_ITEMS              = (3,20)
      START_BYTE              = 3
      AXIS_NAME               = ("XYZ_COMPONENT","TIME"  )
      AXIS_UNIT               = ("N/A"          ,"SECOND")
      AXIS_INTERVAL           = ("N/A"          , 0.2    )
      DESCRIPTION             = "Magnetic field vectors were recorded
                                 at the rate of 10 per second. The
                                 START_TIME field gives the time at
                                 which the first vector in the record
                                 was recorded.  Successive vectors were
                                 recorded  at 0.2 second intervals."

      OBJECT                  = ELEMENT
        NAME                  = MAG_FIELD_COMPONENT_VALUE
        BYTES                 = 2
        DATA_TYPE             = MSB_INTEGER
        START_BYTE            = 1
      END_OBJECT              = ELEMENT
    END_OBJECT                = ARRAY

  END_OBJECT                  = COLLECTION

END_OBJECT                    = ARRAY
END
```

A.3 BIT_COLUMN

The BIT_COLUMN object identifies a string of bits that do not fall on even byte boundaries and therefore cannot be described as a distinct COLUMN. BIT_COLUMNs defined within columns are analogous to columns defined within rows.

Notes:

(1) The Planetary Data System recommends that all fields (within new objects) be defined on byte boundaries. This precludes having multiple values strung together in bit strings, as occurs in the BIT_COLUMN object.

(2) BIT_COLUMN is intended for use in describing existing binary data strings, but is not recommended for use in defining new data objects because it will not be recognized by most general purpose software.

(3) A BIT_COLUMN must not contain embedded objects.

BIT_COLUMNs of the same format and size may be specified as a single BIT_COLUMN by using the ITEMS, ITEM_BITS, and ITEM_OFFSET elements. The ITEMS data element is used to indicate the number of occurrences of a bit string.

A.3.1 Required Keywords

1. NAME
2. BIT_DATA_TYPE
3. START_BIT
4. BITS (required for BIT_COLUMNs without items)
5. DESCRIPTION

A.3.2 Optional Keywords

1. BIT_MASK
2. BITS (optional for BIT_COLUMNSs with ITEMS)
3. FORMAT
4. INVALID_CONSTANT
5. ITEMS
6. ITEM_BITS
7. ITEM_OFFSET
8. MINIMUM
9. MAXIMUM
10. MISSING_CONSTANT
11. OFFSET

12. SCALING_FACTOR
13. UNIT

A.3.3 Required Objects

None

A.3.4 Optional Objects

None

A.3.5 Example

The label fragment below was extracted from a larger example which can be found under the CONTAINER object. The BIT_COLUMN object can be a sub-object only of a COLUMN object, but that COLUMN may itself be part of a TABLE, SPECTRUM, SERIES or CONTAINER object.

```
OBJECT                        = COLUMN
  NAME                        = PACKET_ID
  DATA_TYPE                   = LSB_BIT_STRING
  START_BYTE                  = 1
  BYTES                       = 2
  VALID_MINIMUM               = 0
  VALID_MAXIMUM               = 7
  DESCRIPTION                 = "Packet_id constitutes one of three
                                parts in the primary source information
                                header applied by the Payload Data
                                System (PDS) to the MOLA telemetry
                                packet at the time of creation of the
                                packet prior to transfer frame
                                creation."

  OBJECT                      = BIT_COLUMN
    NAME                      = VERSION_NUMBER
    BIT_DATA_TYPE             = MSB_UNSIGNED_INTEGER
    START_BIT                 = 1
    BITS                      = 3
    MINIMUM                   = 0
    MAXIMUM                   = 7
    DESCRIPTION               = "These bits identify Version 1 as the
                                Source Packet structure.  These bits
                                shall be set to '000'."
  END_OBJECT                  = BIT_COLUMN

  OBJECT                      = BIT_COLUMN
```

```
      NAME                        = SPARE
      BIT_DATA_TYPE               = MSB_UNSIGNED_INTEGER
      START_BIT                   = 4
      BITS                        = 1
      MINIMUM                     = 0
      MAXIMUM                     = 0
      DESCRIPTION                 = "Reserved spare.  This bit shall be set
                                     to '0'"
    END_OBJECT                    = BIT_COLUMN

    OBJECT                        = BIT_COLUMN
      NAME                        = FLAG
      BIT_DATA_TYPE               = BOOLEAN
      START_BIT                   = 5
      BITS                        = 1
      MINIMUM                     = 0
      MAXIMUM                     = 0
      DESCRIPTION                 = "This flag signals the presence or
                                     absence of a Secondary Header data
                                     structure within the Source Packet.
                                     This bit shall be set to '0' since no
                                     Secondary Header formatting standards
                                     currently exist for Mars Observer."
    END_OBJECT                    = BIT_COLUMN

    OBJECT                        = BIT_COLUMN
      NAME                        = ERROR_STATUS
      BIT_DATA_TYPE               = MSB_UNSIGNED_INTEGER
      START_BIT                   = 6
      BITS                        = 3
      MINIMUM                     = 0
      MAXIMUM                     = 7
      DESCRIPTION                 = "This field identifies in part the
                                     individual application process within
                                     the spacecraft that created the Source
                                     Packet data."
    END_OBJECT                    = BIT_COLUMN

    OBJECT                        = BIT_COLUMN
      NAME                        = INSTRUMENT_ID
      BIT_DATA_TYPE               = MSB_UNSIGNED_INTEGER
      START_BIT                   = 9
      BITS                        = 8
      MINIMUM                     = "N/A"
      MAXIMUM                     = "N/A"
      DESCRIPTION                 = "This field identifies in part the
                                     individual application process within
                                     the spacecraft that created the Source
                                     Packet data.  00100011 is the bit
                                     pattern for MOLA."
    END_OBJECT                    = BIT_COLUMN
  END_OBJECT                      = COLUMN
```

A.4 BIT ELEMENT (Primitive Data Object)

Under review.

A.5 CATALOG

The CATALOG object is used within a VOLUME object to reference the completed PDS high-level catalog object set. The catalog object set provides additional information related to the data sets on a volume. Please refer to the *File Specification and Naming* chapter in this document for more information.

A.5.1 Required Keywords

None

A.5.2 Optional Keywords

1. DATA_SET_ID
2. LOGICAL_VOLUME_PATHNAME
3. LOGICAL_VOLUMES

A.5.3 Required Objects

1. DATA_SET
2. INSTRUMENT
3. INSTRUMENT_HOST
4. MISSION

A.5.4 Optional Objects

1. DATA_SET_COLLECTION
2. PERSONNEL
3. REFERENCE
4. TARGET

A.5.5 Example

The example below is a VOLDESC.CAT file for a volume containing multiple data sets. In this case, the catalog objects are provided in separate files referenced by pointers.

```
PDS_VERSION_ID                = PDS3
LABEL_REVISION_NOTE           ="1998-07-01, S. Joy (PPI);"
RECORD_TYPE                   = STREAM

OBJECT                        = VOLUME
```

```
VOLUME_SERIES_NAME          = "VOYAGERS TO THE OUTER PLANETS"
VOLUME_SET_NAME             = "VOYAGER NEPTUNE PLANETARY PLASMA
                               INTERACTIONS DATA"
VOLUME_SET_ID               = USA_NASA_PDS_VG_1001
VOLUMES                     = 1
VOLUME_NAME                 = "VOYAGER NEPTUNE PLANETARY PLASMA
                               INTERACTIONS DATA"
VOLUME_ID                   = VG_1001
VOLUME_VERSION_ID           = "VERSION 1"
VOLUME_FORMAT               = "ISO-9660"
MEDIUM_TYPE                 = "CD-ROM"
PUBLICATION_DATE            = 1992-11-13
DESCRIPTION                 = "This volume contains a collection of
                               non-imaging Planetary Plasma datasets
                               from the Voyager 2 spacecraft encounter
                               with Neptune.  Included are datasets
                               from the Cosmic Ray System (CRS),
                               Plasma System (PLS), Plasma Wave System
                               (PWS), Planetary Radio Astronomy (PRA),
                               Magnetometer (MAG), and Low Energy
                               Charged Particle (LECP) instruments, as
                               well as spacecraft position vectors
                               (POS) in several coordinate systems.
                               The volume also contains documentation
                               and index files to support access and
                               use of the data."

DATA_SET_ID                 = {"VG2-N-CRS-3-RDR-D1-6SEC-V1.0",
                                "VG2-N-CRS-4-SUMM-D1-96SEC-V1.0",
                                "VG2-N-CRS-4-SUMM-D2-96SEC-V1.0",
                                "VG2-N-LECP-4-SUMM-SCAN-24SEC-V1.0",
                                "VG2-N-LECP-4-RDR-STEP-12.8MIN-V1.0",
                                "VG2-N-MAG-4-RDR-HG-COORDS-1.92SEC-V1.0",
                                "VG2-N-MAG-4-SUMM-HG-COORDS-48SEC-V1.0",
                                "VG2-N-MAG-4-RDR-HG-COORDS-9.6SEC-V1.0",
                                "VG2-N-MAG-4-SUMM-NLSCOORDS-12SEC-V1.0",
                                "VG2-N-PLS-5-RDR-2PROMAGSPH-48SEC-V1.0",
                                "VG2-N-PLS-5-RDR-ELEMAGSPHERE-96SEC-V1.0",
                                "VG2-N-PLS-5-RDR-IONMAGSPHERE-48SEC-V1.0",
                                "VG2-N-PLS-5-RDR-IONLMODE-48SEC-V1.0",
                                "VG2-N-PLS-5-RDR-IONMMODE-12MIN-V1.0",
                                "VG2-N-PLS-5-RDR-ION-INBNDWIND-48SEC-V1.0",
                                "VG2-N-POS-5-RDR-HGHGCOORDS-48SEC-V1.0",
                                "VG2-N-POS-5-SUMM-NLSCOORDS-12-48SEC-V1.0",
                                "VG2-N-PRA-4-SUMM-BROWSE-SEC-V1.0",
                                "VG2-N-PRA-2-RDR-HIGHRATE-60MS-V1.0",
                                "VG2-N-PWS-2-RDR-SA-4SEC-V1.0",
                                "VG2-N-PWS-4-SUMM-SA-48SEC-V1.0",
                                "VG2-N-PWS-1-EDR-WFRM-60MS-V1.0"}

  OBJECT                    = DATA_PRODUCER
    INSTITUTION_NAME        = "UNIVERSITY OF CALIFORNIA, LOS ANGELES"
    FACILITY_NAME           = "PDS PLANETARY PLASMA INTERACTIONS
                               NODE"
```

```
      FULL_NAME                   = "DR. RAYMOND WALKER"
      DISCIPLINE_NAME             = "PLASMA INTERACTIONS"
      ADDRESS_TEXT                = "UCLA
                                     IGPP
                                     LOS ANGELES, CA 90024  USA"
    END_OBJECT                    = DATA_PRODUCER

    OBJECT                        = DATA_SUPPLIER
      INSTITUTION_NAME            = "NATIONAL SPACE SCIENCE DATA CENTER"
      FACILITY_NAME               = "NATIONAL SPACE SCIENCE DATA CENTER"
      FULL_NAME                   = "NATIONAL SPACE SCIENCE DATA CENTER"
      DISCIPLINE_NAME             = "NATIONAL SPACE SCIENCE DATA CENTER"
      ADDRESS_TEXT                = "Code 633  \n
                                     Goddard Space Flight Center  \n
                                     Greenbelt, Maryland, 20771, USA"
      TELEPHONE_NUMBER            = "3012866695"
      ELECTRONIC_MAIL_TYPE        = "NSI/DECNET"
      ELECTRONIC_MAIL_ID          = "NSSDCA::REQUEST"
    END_OBJECT                    = DATA_SUPPLIER

    OBJECT                        = CATALOG
      ^MISSION_CATALOG            = "MISSION.CAT"
      ^INSTRUMENT_HOST_CATALOG    = "INSTHOST.CAT"
      ^INSTRUMENT_CATALOG         = {"CRS_INST.CAT",
                                     "LECPINST.CAT",
                                     "MAG_INST.CAT",
                                     "PLS_INST.CAT",
                                     "PRA_INST.CAT",
                                     "PWS_INST.CAT"}
      ^DATA_SET_CATALOG           = {"CRS_DS.CAT",
                                     "LECP_DS.CAT",
                                     "MAG_DS.CAT",
                                     "PLS_DS.CAT",
                                     "POS_DS.CAT",
                                     "PRA_DS.CAT",
                                     "PWS_DS.CAT"}
      ^TARGET_CATALOG             = TARGET.CAT
      ^PERSONNEL_CATALOG          = PERSON.CAT
      ^REFERENCE_CATALOG          = REF.CAT
    END_OBJECT                    = CATALOG

  END_OBJECT                      = VOLUME
  END
```

A.6 COLLECTION (Primitive Data Object)

The COLLECTION object allows the ordered grouping of heterogeneous objects into a structure. The COLLECTION object may contain a mixture of different object types, including other COLLECTIONs. The optional START_BYTE data element provides the starting location relative to an enclosing object. If a START_BYTE is not specified, a value of 1 is assumed.

A.6.1 Required Keywords

1. BYTES
2. NAME

A.6.2 Optional Keywords

1. DESCRIPTION
2. CHECKSUM
3. INTERCHANGE_FORMAT
4. START_BYTE

A.6.3 Required Objects

None

Note that although a specific sub-object is not required, the COLLECTION must contain at least one of the optional objects listed following. That is, a null COLLECTION may not be defined.

A.6.4 Optional Objects

1. ELEMENT
2. BIT_ELEMENT
3. ARRAY
4. COLLECTION

A.6.5 Example

Please refer to Section A.2.6, *Example 2* under the ARRAY object for an illustration of the COLLECTION object used in conjunction with other primitive objects.

A.7 COLUMN

The COLUMN object identifies a single column in a data object.

Notes:

(1) Current PDS data objects that include COLUMN objects are the TABLE, CONTAINER, SPECTRUM and SERIES objects.

(2) COLUMNs must not themselves contain embedded COLUMN objects.

(3) COLUMNs of the same format and size which constitute a vector may be specified as a single COLUMN by using the ITEMS, ITEM_BYTES, and ITEM_OFFSET elements. The ITEMS data element indicates the number of occurrences of the field (i.e., elements in the vector).

(4) BYTES and ITEM_BYTES counts do not include leading or trailing delimiters or line terminators.

(5) For a COLUMN containing ITEMS, the value of BYTES should represent the total size of the column including delimiters between the items. (See examples 1 and 2 below.)

A.7.1 Required Keywords

1. NAME
2. DATA_TYPE
3. START_BYTE
4. BYTES (required for COLUMNs without ITEMs)

A.7.2 Optional Keywords

1. BIT_MASK
2. BYTES (optional for COLUMNs with ITEMs)
3. COLUMN_NUMBER
4. DERIVED_MAXIMUM
5. DERIVED_MINIMUM
6. DESCRIPTION
7. FORMAT
8. INVALID_CONSTANT
9. ITEM_BYTES
10. ITEM_OFFSET
11. ITEMS
12. MAXIMUM
13. MAXIMUM_SAMPLING_PARAMETER
14. MINIMUM
15. MINIMUM_SAMPLING_PARAMETER

16. MISSING_CONSTANT
17. OFFSET
18. SAMPLING_PARAMETER_INTERVAL
19. SAMPLING_PARAMETER_NAME
20. SAMPLING_PARAMETER_UNIT
21. SCALING_FACTOR
22. UNIT
23. VALID_MAXIMUM
24. VALID_MINIMUM

A.7.3 Required Objects

None

A.7.4 Optional Objects

1. BIT_COLUMN
2. ALIAS

A.7.5 Example 1

The label fragment below shows a simple COLUMN object, in this case from an ASCII TABLE.

```
OBJECT                        = COLUMN
  NAME                        = "DETECTOR TEMPERATURE"
  START_BYTE                  = 27
  BYTES                       = 5
  DATA_TYPE                   = ASCII_REAL
  FORMAT                      = "F5.1"
  UNIT                        = "KELVIN"
  MISSING_CONSTANT            = 999.9
END_OBJECT                    = COLUMN
```

A.7.6 Example 2

The fragment below shows two COLUMNs containing multiple items. The first COLUMN is a vector containing three ASCII_INTEGER items: xx, yy, zz. The second COLUMN contains three character items: “xx”, “yy” and “zz”. Note that the value of BYTES includes the comma delimiters between items, but the ITEM_BYTES value does not. The ITEM_OFFSET is the number of bytes from the beginning of one item to the beginning of the next.

```
OBJECT                  = COLUMN
  NAME                  = COLUMN1XYZ
  DATA_TYPE             = ASCII_INTEGER
```

```
    START_BYTE              = 1
    BYTES                   = 8  /*includes delimiters*/
    ITEMS                   = 3
    ITEM_BYTES              = 2
    ITEM_OFFSET             = 3
  END_OBJECT                = COLUMN

  OBJECT                    = COLUMN
    NAME                    = COLUMN2XYZ
    DATA_TYPE               = CHARACTER
    START_BYTE              = 2   /* value does not include leading quote
                                     */
    BYTES                   = 12  /* value does not include leading and
                                     */
                                  /*  trailing quotes */
    ITEMS                   = 3
    ITEM_BYTES              = 2   /* value does not include leading and
                                     */
                                  /*  trailing quotes */
    ITEM_OFFSET             = 5   /* value does not include leading quote
                                     */
  END_OBJECT                = COLUMN
```

A.7.7 Example 3

The fragment below was extracted from a larger example which can be found under the CONTAINER object. It illustrates a single COLUMN object subdivided into several BIT_COLUMN fields.

```
  OBJECT                          = COLUMN
    NAME                          = PACKET_ID
    DATA_TYPE                     = LSB_BIT_STRING
    START_BYTE                    = 1
    BYTES                         = 2
    VALID_MINIMUM                 = 0
    VALID_MAXIMUM                 = 7
    DESCRIPTION                   = "Packet_id constitutes one of three
                                     parts in the  primary source
                                     information header applied by the
                                     Payload Data System (PDS) to the MOLA
                                     telemetry packet at the time of
                                     creation of the packet prior to
                                     transfer frame creation.  "

    OBJECT                        = BIT_COLUMN
      NAME                        = VERSION_NUMBER
      BIT_DATA_TYPE               = MSB_UNSIGNED_INTEGER
      START_BIT                   = 1
      BITS                        = 3
      MINIMUM                     = 0
      MAXIMUM                     = 7
      DESCRIPTION                 = "These bits identify Version 1 as the
                                     Source Packet structure.  These bits
                                     shall be set to '000'."
```

```
    END_OBJECT                   = BIT_COLUMN


    OBJECT                       = BIT_COLUMN
      NAME                       = SPARE
      BIT_DATA_TYPE              = MSB_UNSIGNED_INTEGER
      START_BIT                  = 4
      BITS                       = 1
      MINIMUM                    = 0
      MAXIMUM                    = 0
      DESCRIPTION                = "Reserved spare.  This bit shall be set
                                    to '0'"
    END_OBJECT                   = BIT_COLUMN

    OBJECT                       = BIT_COLUMN
      NAME                       = FLAG
      BIT_DATA_TYPE              = BOOLEAN
      START_BIT                  = 5
      BITS                       = 1
      MINIMUM                    = 0
      MAXIMUM                    = 0
      DESCRIPTION                = "This flag signals the presence or
                                    absence of a Secondary Header data
                                    structure within the Source Packet.
                                    This bit shall be set to '0' since no
                                    Secondary Header formatting standards
                                    currently exist for Mars Observer."
    END_OBJECT                   = BIT_COLUMN

    OBJECT                       = BIT_COLUMN
      NAME                       = ERROR_STATUS
      BIT_DATA_TYPE              = MSB_UNSIGNED_INTEGER
      START_BIT                  = 6
      BITS                       = 3
      MINIMUM                    = 0
      MAXIMUM                    = 7
      DESCRIPTION                = "This field identifies in part the
                                    individual application process within
                                    the spacecraft that created the Source
                                    Packet data."
    END_OBJECT                   = BIT_COLUMN

    OBJECT                       = BIT_COLUMN
      NAME                       = INSTRUMENT_ID
      BIT_DATA_TYPE              = MSB_UNSIGNED_INTEGER
      START_BIT                  = 9
      BITS                       = 8
      MINIMUM                    = "N/A"
      MAXIMUM                    = "N/A"
      DESCRIPTION                = "This field identifies in part the
                                    individual application process within
                                    the spacecraft that creeated the Source
                                    Packet data.  00100011 is the bit
                                    pattern for MOLA."
    END_OBJECT                   = BIT_COLUMN
  END_OBJECT                     = COLUMN
```

A.8 CONTAINER

The CONTAINER object is used to group a set of sub-objects (such as COLUMNs) that repeat within a data object (such as a TABLE). Use of the CONTAINER object allows repeating groups to be defined within a data structure.

A.8.1 Required Keywords

1. NAME
2. START_BYTE
3. BYTES
4. REPETITIONS
5. DESCRIPTION

A.8.2 Optional Keywords

Any

A.8.3 Required Objects

None

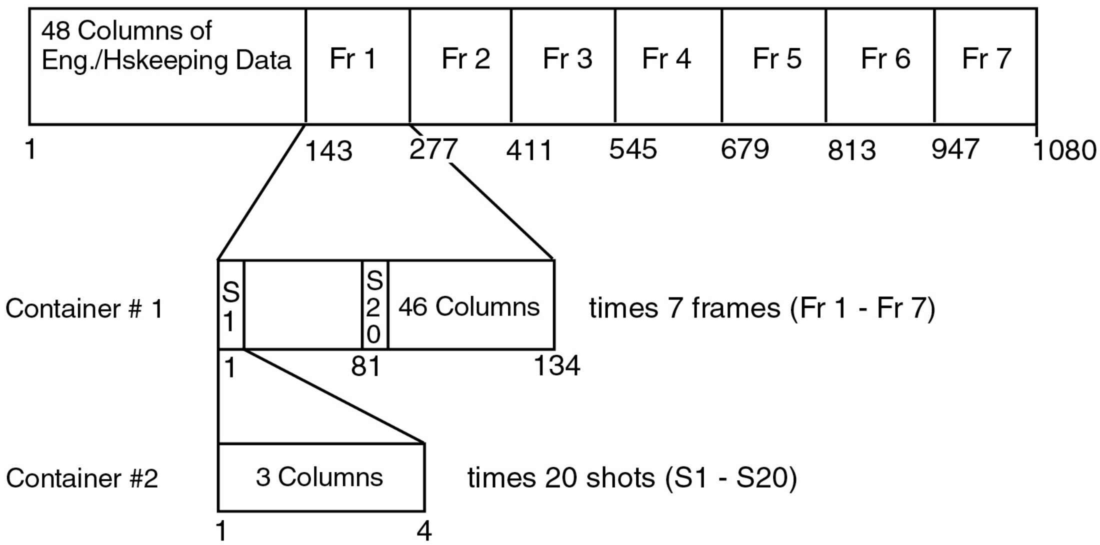
A.8.4 Optional Objects

1. COLUMN
2. CONTAINER

A.8.5 Example

The set of labels and format fragments below illustrates a data product layout in which the CONTAINER object is used. The primary data product is a TABLE of data records. Each record within the TABLE begins with 48 columns (143 bytes) of engineering data. The data product acquires science data from seven different frames. Since the data from each frame are formatted identically, one CONTAINER description suffices for all seven frames.

In this example there are two CONTAINER objects. The first CONTAINER object describes the repeating frame information. Within this CONTAINER there is a second CONTAINER object in which a 4-byte set of three COLUMN objects repeats 20 times. The use of the second CONTAINER object permits the data supplier to describe the three COLUMNS (4 bytes) once, instead of specifying sixty column definitions.



In the first CONTAINER, the keyword REPETITIONS is equal to 7. In the second CONTAINER, REPETITIONS equals 20. Both CONTAINER objects contain a collection of COLUMN objects. In most cases it is preferable to save space in the product label by placing COLUMN objects in a separate file and pointing to that file from within the CONTAINER object.

This attached label example describes the above TABLE structure using CONTAINER objects.

```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
FILE_RECORDS                    = 467
RECORD_BYTES                    = 1080
LABEL_RECORDS                   = 4
FILE_NAME                       = "AEDR.A01"

^MOLA_SCIENCE_MODE_TABLE        = 5
DATA_SET_ID                     = "MO-M-MOLA-1-AEDR-L0-V1.0"
PRODUCT_ID                      = "MOLA-AEDR-10010-0001"
SPACECRAFT_NAME                 = MARS_OBSERVER
INSTRUMENT_ID                   = MOLA
INSTRUMENT_NAME                 = MARS_OBSERVER_LASER_ALTIMETER
TARGET_NAME                     = MARS
SOFTWARE_NAME                   = "BROWSER 17.1"
UPLOAD_ID                       = "5.3"
PRODUCT_RELEASE_DATE            = 1994-12-29
START_TIME                      = 1994-09-29T04:12:43.983
STOP_TIME                       = 1994-09-29T06:09:54.221
SPACECRAFT_CLOCK_START_COUNT    = "12345"
SPACECRAFT_CLOCK_STOP_COUNT     = "12447"
PRODUCT_CREATION_TIME           = 1994-01-29T07:30:333
MISSION_PHASE_NAME              = MAPPING
```

```
ORBIT_NUMBER                  = 0001
PRODUCER_ID                   = MO_MOLA_TEAM
PRODUCER_FULL_NAME            = "DAVID E. SMITH"
PRODUCER_INSTITUTION_NAME     = "GODDARD SPACE FLIGHT CENTER"
DESCRIPTION                   = "This data product contains the
      aggregation of MOLA telemetry packets by Orbit.  All Experiment
      Data Record Packets retrieved from the PDB are collected in this
      data product.  The AEDR data product is put together with the
      Project-provided software tool Browser."

OBJECT                        = MOLA_SCIENCE_MODE_TABLE
  INTERCHANGE_FORMAT          = BINARY
  ROWS                        = 463
  COLUMNS                     = 97
  ROW_BYTES                   = 1080
  ^STRUCTURE                  = "MOLASCI.FMT"
  DESCRIPTION                 = "This table is one of two that describe
      the arrangement of information on the Mars Observer Laser
      Altimeter (MOLA) Aggregated Engineering Data Record (AEDR).  ..."

END_OBJECT                    = MOLA_SCIENCE_MODE_TABLE
...

END
```

Contents of the MOLASCI.FMT file:

```
OBJECT                        = COLUMN
  NAME                        = PACKET_ID
  DATA_TYPE                   = LSB_BIT_STRING
  START_BYTE                  = 1
  BYTES                       = 2
  VALID_MINIMUM               = 0
  VALID_MAXIMUM               = 7
  DESCRIPTION                 = "Packet_id constitutes one of three
      parts in the  primary source information header applied by the
      Payload Data System (PDS) to the MOLA telemetry packet at the
      time of creation of the packet prior to transfer frame creation."

  OBJECT                      = BIT_COLUMN
    NAME                      = VERSION_NUMBER
    BIT_DATA_TYPE             = UNSIGNED_INTEGER
    START_BIT                 = 1
    BITS                      = 3
    MINIMUM                   = 0
    MAXIMUM                   = 7
    DESCRIPTION               = "These bits identify Version 1 as the
      Source Packet structure.  These bits shall be set to '000'."
  END_OBJECT                  = BIT_COLUMN

  OBJECT                      = BIT_COLUMN
    NAME                      = SPARE
    BIT_DATA_TYPE             = UNSIGNED_INTEGER
```

```
    START_BIT                  = 4
    BITS                       = 1
    MINIMUM                    = 0
    MAXIMUM                    = 0
    DESCRIPTION                = "Reserved spare.  This bit shall be set
                                 to '0'"
  END_OBJECT                   = BIT_COLUMN

  OBJECT                       = BIT_COLUMN
    NAME                       = SECONDARY_HEADER_FLAG
    BIT_DATA_TYPE              = BOOLEAN
    START_BIT                  = 5
    BITS                       = 1
    MINIMUM                    = 0
    MAXIMUM                    = 0
    DESCRIPTION                = "This flag signals the presence or
      absence of a Secondary Header data structure within the Source
      Packet. This bit shall be set to '0' since no Secondary Header
      formatting standards currently exist for Mars Observer."
  END_OBJECT                   = BIT_COLUMN

  OBJECT                       = BIT_COLUMN
    NAME                       = ERROR_STATUS
    BIT_DATA_TYPE              = UNSIGNED_INTEGER
    START_BIT                  = 6
    BITS                       = 3
    MINIMUM                    = 0
    MAXIMUM                    = 7
    DESCRIPTION                = "This field identifies in part the
      individual application process within the spacecraft that created
      the Source Packet data."
  END_OBJECT                   = BIT_COLUMN

  OBJECT                       = BIT_COLUMN
    NAME                       = INSTRUMENT_ID
    BIT_DATA_TYPE              = UNSIGNED_INTEGER
    START_BIT                  = 9
    BITS                       = 8
    MINIMUM                    = 2#0100011#
    MAXIMUM                    = 2#0100011#
    DESCRIPTION                = "This field identifies in part the
      individual application process within the spacecraft that created
      the Source Packet data.  00100011 is the bit pattern for MOLA."
  END_OBJECT                   = BIT_COLUMN
END_OBJECT                     = COLUMN

...

OBJECT                   = COLUMN
  NAME                         = COMMAND_ECHO
  DATA_TYPE                    = INTEGER
  START_BYTE                   = 125
  BYTES                        = 16
  ITEMS                        = 8
```

```
  ITEM_BYTES                = 2
  MINIMUM                   = 0
  MAXIMUM                   = 65535
  DESCRIPTION               = "First 8 command words received during
      current packet, only complete commands are stored, MOLA specific
      commands only. The software attempts to echo all valid commands.
      If the command will fit in the room remaining in the..."
END_OBJECT                  = COLUMN

OBJECT                      = COLUMN
  NAME                      = PACKET_VALIDITY_CHECKSUM
  DATA TYPE                 = INTEGER
  START_BYTE                = 141
  BYTES                     = 2
  MINIMUM                   = 0
  MAXIMUM                   = 65535
  DESCRIPTION               = "Simple 16 bit addition of entire
      packet contents upon completion.  This location is zeroed for
      addition. This word is zeroed, then words 0-539 are added without
      carry to a variable that is initially zero. The resulting lower
      16 bits are..."
END_OBJECT                  = COLUMN

OBJECT                      = CONTAINER
  NAME                      = FRAME_STRUCTURE
  ^STRUCTURE                = "MOLASCFR.FMT" /*points to the
                              columns*/
                               /*that make up the frame descriptors
                              */
  START_BYTE                = 143
  BYTES                     = 134
  REPETITIONS               = 7
  DESCRIPTION               = "The frame_structure container
      represents the format of seven repeating groups of attributes in
      this data product.  The data product reflects science data
      acquisition from seven different frames.  Since the data from
      each frame are ..."
END_OBJECT                  = CONTAINER
```

Contents of the MOLASCFR.FMT FILE:

```
OBJECT                      = CONTAINER
  NAME                      = COUNTS
  START_BYTE                = 1
  BYTES                     = 4
  REPETITIONS               = 20
   ^STRUCTURE               = "MOLASCCT.FMT"
  DESCRIPTION               = "This container has three sub-elements
      (range to surface counts, 1st channel received pulse energy, and
      2nd channel received pulse energy).  The three sub-elements
      repeat for each of 20 shots."
END_OBJECT                  = CONTAINER
```

```
OBJECT                         = COLUMN
  NAME                         = SHOT_2_LASER_TRANSMITTER_POWR
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 81
  BYTES                        = 1
  MINIMUM                      = 0
  MAXIMUM                      = 65535
  DESCRIPTION                  = "..."
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = SHOT_1_LASER_TRANSMITTER_POWR
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 82
  BYTES                        = 1
  MINIMUM                      = 0
  MAXIMUM                      = 65535
  DESCRIPTION                  = "..."
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = SHOT_4_LASER_TRANSMITTER_POWR
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 83
  BYTES                        = 1
  MINIMUM                      = 0
  MAXIMUM                      = 65535
  DESCRIPTION                  = "..."
END_OBJECT                     = COLUMN

...

OBJECT                         = COLUMN
  NAME                         = CH_3_2ND_HALF_FRAME_BKGRND_CN
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 133
  BYTES                        = 1
  MINIMUM                      = 0
  MAXIMUM                      = 255
  DESCRIPTION                  = "The background energy or noise count
      levels in channels 1, 2, 3, and 4 respectively by half-frame.
      Pseudo log value of NOISE(1, 2, 3, 4) at the end of a half-frame
      of current frame, 5.3 bit format.  Plog base 2 of background
      count sum..."
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = CH_4_2ND_HALF_FRAME_BKGRND_CN
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 134
  BYTES                        = 1
  MINIMUM                      = 0
  MAXIMUM                      = 255
```

```
  DESCRIPTION                  = "The background energy or noise count
      levels in channels 1, 2, 3, and 4 respectively by half-frame.
      Pseudo log value of NOISE(1, 2, 3, 4) at the end of a half-frame
      of current frame, 5.3 bit format.  Plog base 2 of background
      count sum..."
END_OBJECT                     = COLUMN
```

Contents of the MOLASCCT.FMT FILE:

```
OBJECT                         = COLUMN
  NAME                         = RANGE_TO_SURFACE_TIU_CNTS
  DATA_TYPE                    = MSB_INTEGER
  START_BYTE                   = 1
  BYTES                        = 2
  DESCRIPTION                  = "The possible 20 valid frame laser
      shots surface ranging measurements in Timing Interval Unit (TIU)
      counts.  The least significant 16 bits of TIU (SLTIU), stored for
      every shot. B[0] = Bits 15-8 of TIU reading; B[1] = Bits 7-0 of
      ..."
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = FIRST_CH_RCVD_PULSE_ENRGY
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 3
  BYTES                        = 1
  DESCRIPTION                  = "The level of return, reflected energy
      as received by the first channel and matched filter to trigger.
      This is a set of  values for all possible 20 shots within the
      frame.  Lowest numbered non-zero energy reading for each shot."
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = SECOND_CH_RCVD_PULSE_ENRGY
  DATA_TYPE                    = UNSIGNED_INTEGER
  START_BYTE                   = 4
  BYTES                        = 1
  DESCRIPTION                  = "The level of return, reflected energy
      as received by the second channel and matched filter to trigger.
      This is a set of values for all possible 20 shots within the
      frame.  2nd lowest numbered non-zero energy reading for each
      shot..."
END_OBJECT                     = COLUMN
```

A.9 DATA_PRODUCER

The DATA_PRODUCER object is a required sub-object of the VOLUME object. The DATA_PRODUCER, as opposed to the DATA_SUPPLIER, is an individual or organization responsible for collecting, assembling, and/or engineering the raw data into one or more data sets.

A.9.1 Required Keywords

1. INSTITUTION_NAME
2. FACILITY_NAME
3. FULL_NAME
4. ADDRESS_TEXT

A.9.2 Optional Keywords

1. DISCIPLINE_NAME
2. NODE_NAME
3. TELEPHONE_NUMBER
4. ELECTRONIC_MAIL_TYPE
5. ELECTRONIC_MAIL_ID

A.9.3 Required Objects

None

A.9.4 Optional Objects

None

A.9.5 Example

The fragment below was extracted from the example under the VOLUME object.

```
OBJECT                        = DATA_PRODUCER
  INSTITUTION_NAME            = "U.S.G.S. FLAGSTAFF"
  FACILITY_NAME               = "BRANCH OF ASTROGEOLOGY"
  FULL_NAME                   = "ERIC M. ELIASON"
  DISCIPLINE_NAME             = "IMAGE PROCESSING"
  ADDRESS_TEXT                = "Branch of Astrogeology
                                 United States Geological Survey
                                 2255 North Gemini Drive
```

```
                                    Flagstaff, Arizona 86001 USA"
      END_OBJECT                  = DATA_PRODUCER
```

A.10 DATA_SUPPLIER

The DATA_SUPPLIER object is an optional sub-object of the VOLUME object. The DATA_SUPPLIER, as opposed to the DATA_PRODUCER, is an individual or organization responsible for distributing the data sets and associated data to the science community.

A.10.1 Required Keywords

1. INSTITUTION_NAME
2. FACILITY_NAME
3. FULL_NAME
4. ADDRESS_TEXT
5. TELEPHONE_NUMBER
6. ELECTRONIC_MAIL_TYPE
7. ELECTRONIC_MAIL_ID

A.10.2 Optional Keywords

1. DISCIPLINE_NAME
2. NODE_NAME

A.10.3 Required Objects

None

A.10.4 Optional Objects

None

A.10.5 Example

The fragment below was extracted from the larger example which can be found under the VOLUME object.

```
OBJECT                         = DATA_SUPPLIER
  INSTITUTION_NAME             = "NATIONAL SPACE SCIENCE DATA CENTER"
  FACILITY_NAME                = "NATIONAL SPACE SCIENCE DATA CENTER"
  FULL_NAME                    = "NATIONAL SPACE SCIENCE DATA CENTER"
  DISCIPLINE_NAME              = "NATIONAL SPACE SCIENCE DATA CENTER"
  ADDRESS_TEXT                 = "Code 633
                                  Goddard Space Flight Center
                                  Greenbelt, Maryland, 20771, USA"
  TELEPHONE_NUMBER             = "3012866695"
  ELECTRONIC_MAIL_TYPE         = "NSI/DECNET"
```

```
  ELECTRONIC_MAIL_ID           = "NSSDCA::REQUEST"
END_OBJECT                     = DATA_SUPPLIER
```

A.11 DIRECTORY

The DIRECTORY object is used to define a hierarchical file organization on a linear (i.e., sequential) medium such as tape. The DIRECTORY object identifies all directories and subdirectories below the root level. It is a required sub-object of the VOLUME object for volumes delivered on sequential media.

Note: The root directory on a volume does not need to be explicitly defined with the DIRECTORY object.

Subdirectories are identified by defining DIRECTORY objects as sub-objects of the root DIRECTORY. Files within the directories and subdirectories are sequentially identified by using FILE objects with a SEQUENCE_NUMBER value corresponding to their position on the medium. The SEQUENCE_NUMBER value must be unique for each file on the medium.

A.11.1 Required Keywords

1. NAME

A.11.2 Optional Keywords

1. RECORD_TYPE
2. SEQUENCE_NUMBER

A.11.3 Required Objects

1. FILE

A.11.4 Optional Objects

1. DIRECTORY

A.11.5 Example

The fragment below was extracted from the larger example which can be found under the VOLUME object.

```
OBJECT                        = DIRECTORY
  NAME                        = INDEX

  OBJECT                      = FILE
    FILE_NAME                 = "INDXINFO.TXT"
    RECORD_TYPE               = STREAM
    SEQUENCE_NUMBER           = 5
  END_OBJECT                  = FILE

  OBJECT                      = FILE
    FILE_NAME                 = "INDEX.LBL"
    RECORD_TYPE               = STREAM
    SEQUENCE_NUMBER           = 6
  END_OBJECT                  = FILE

  OBJECT                      = FILE
    FILE_NAME                 = "INDEX.TAB"
    RECORD_TYPE               = FIXED_LENGTH
    RECORD_BYTES              = 512
    FILE_RECORDS              = 6822
    SEQUENCE_NUMBER           = 7
  END_OBJECT                  = FILE
END_OBJECT                    = DIRECTORY
```

A.12 DOCUMENT

> ***Note:*** *This section is currently undergoing major revision. Please consult a PDS data engineer for the latest available information on document labelling.*

The DOCUMENT object is used to label a particular document that is provided on a volume to support an archived data product. A document can be made up of one or more files in a single format. For instance, a document may be comprised of as many TIFF files as there are pages in the document.

Multiple versions of a document can be supplied on a volume with separate formats, requiring a DOCUMENT object for each document version (i.e., OBJECT = TEX_DOCUMENT and OBJECT = PS_DOCUMENT when including both the TEX and Postscript versions of the same document).

PDS requires that at least one version of any document be plain ASCII text in order to allow users the capability to read, browse, or search the text without requiring software or text processing packages. This version can be plain, unmarked text, or ASCII text containing a markup language. (See the *Documentation* chapter of this document for more details.)

The DOCUMENT object contains keywords that identify and describe the document, provide the date of publication of the document, indicate the number of files comprising the document, provide the format of the document files, and identify the software used to compress or encode the document, as applicable.

DOCUMENT labels must be detached files unless the files are plain, unmarked text that will not be read by text or word processing packages. A DOCUMENT object for each format type of a document can be included in the same label file with pointers, such as ^TIFF_DOCUMENT for a TIFF formatted document. (See example below.)

A.12.1 Required Keywords

1. DOCUMENT_NAME
2. DOCUMENT_TOPIC_TYPE
3. INTERCHANGE_FORMAT
4. DOCUMENT_FORMAT
5. PUBLICATION_DATE

A.12.2 Optional Keywords

1. ABSTRACT_TEXT
2. DESCRIPTION
3. ENCODING_TYPE

4. FILES

A.12.3 Required Objects

None

A.12.4 Optional Objects

None

A.12.5 Example

The following example detached label, PDSUG.LBL, is for a Document provided in three formats: ASCII text, TIFF, and TEX.

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = UNDEFINED

^ASCII_DOCUMENT               = "PDSUG.ASC"
^TIFF_DOCUMENT                = {"PDSUG001.TIF", "PDSUG002.TIF",
                                 "PDSUG003.TIF", "PDSUG004.TIF" }
^TEX_DOCUMENT                 = "PDSUG.TEX"

OBJECT                        = ASCII_DOCUMENT
  DOCUMENT_NAME               = "Planetary Data System Data Set Catalog
                                 User's Guide"
  PUBLICATION_DATE            = 1992-04-13
  DOCUMENT_TOPIC_TYPE         = "USER'S GUIDE"
  INTERCHANGE_FORMAT          = ASCII
  DOCUMENT_FORMAT             = TEXT

  DESCRIPTION                 = "The Planetary Data System Data Set
    Catalog User's Guide describes the fundamentals of accessing,
    searching, browsing, and ordering data from the PDS Data Set Catalog
    at the Central Node.  The text for this 4-page document is provided
    here in this plain, ASCII text file."
  ABSTRACT_TEXT               = "The PDS Data Set Catalog is similar in
    function and purpose to a card catalog in a library.  Use a Search
    screen to find data items, a List/Order screen to order data items,
    and the More menu option to see more information."
END_OBJECT                    = ASCII_DOCUMENT

OBJECT                        = TIFF_DOCUMENT
  DOCUMENT_NAME               = "Planetary Data System Data Set Catalog
                                 User's Guide"
  DOCUMENT_TOPIC_TYPE         = "USER'S GUIDE"
  INTERCHANGE_FORMAT          = BINARY
```

```
    DOCUMENT_FORMAT              = TIFF
    PUBLICATION_DATE             = 1992-04-13
    FILES                        = 4
    ENCODING_TYPE                = "CCITT/3"
    DESCRIPTION                  = "The Planetary Data System Data Set
      Catalog User's Guide describes the fundamentals of accessing,
      searching, browsing, and ordering data from the PDS Data Set Catalog
      at the Central Node.

      The 4-page document is provided here in 4 consecutive files, one
      file per page, in Tagged Image File Format (TIFF) using Group 3
      compression.  It has been successfully imported into WordPerfect
      5.0, FrameMaker, and Photoshop."
    ABSTRACT_TEXT                = "The PDS Data Set Catalog is similar in
      function and purpose to a card catalog in a library.  Use a Search
      screen to find data items, a List/Order screen to order data items,
      and the More menu option to see more information."
  END_OBJECT                     = TIFF_DOCUMENT

  OBJECT                         = TEX_DOCUMENT
    DOCUMENT_NAME                = "Planetary Data System Data Set Catalog
                                    User's Guide"
    DOCUMENT_TOPIC_TYPE          = "USER'S GUIDE"
    INTERCHANGE_FORMAT           = ASCII
    DOCUMENT_FORMAT              = TEX
    PUBLICATION_DATE             = 1992-04-13
    DESCRIPTION                  = "The Planetary Data System Data Set
      Catalog User's Guide describes the fundamentals of accessing,
      searching, browsing, and ordering data from the PDS Data Set Catalog
      at the Central Node.

      The 4-page document is provided here in TeX format with all
      necessary macros included."
    ABSTRACT_TEXT                = "The PDS Data Set Catalog is similar in
      function and purpose to a card catalog in a library.  Use a Search
      screen to find data items, a List/Order screen to order data items,
      and the More menu option to see more information."
  END_OBJECT                     = TEX_DOCUMENT
  END
```

A.13 ELEMENT (Primitive Data Object)

The ELEMENT object provides a means of defining a lowest-level component of a data object, and which can be stored in an integral multiple of 8-bit bytes. ELEMENT objects may be embedded in COLLECTION and ARRAY data objects. The optional START_BYTE element identifies a location relative to the enclosing object. If not explicitly included, a START_BYTE = 1 is assumed for the ELEMENT.

A.13.1 Required Keywords

1. BYTES
2. DATA_TYPE
3. NAME

A.13.2 Optional Keywords

1. START_BYTE
2. BIT_MASK
3. DERIVED_MAXIMUM
4. DERIVED_MINIMUM
5. DESCRIPTION
6. FORMAT
7. INVALID_CONSTANT
8. MINIMUM
9. MAXIMUM
10. MISSING_CONSTANT
11. OFFSET
12. SCALING_FACTOR
13. UNIT
14. VALID_MINIMUM
15. VALID_MAXIMUM

A.13.3 Required Objects

None

A.13.4 Optional Objects

None

A.13.5 Example

Please refer to the example in the ARRAY Primitive object (Section A.2) for an example of the use of the ELEMENT object.

A.14 FIELD

The FIELD object identifies a single variable-width field in a SPREADSHEET object.

Notes:

1. The only PDS data object that includes FIELD objects is the SPREADSHEET. FIELDs must not themselves contain embedded FIELD objects.

2. The DATA_TYPE keyword is required to specify the data type of the values that are stored in the field when data are present.

3. A vector with two or more identically formatted components may be specified as a single FIELD by using the ITEM and ITEM_BYTES elements. The ITEMS data element indicates the number of occurrences within the field (i.e., components in the vector).

4. If a FIELD contains multiple items, then the ITEM_BYTES keyword is used to specify the maximum number of bytes any item in the set may have. ITEM_BYTES does not include the quotation marks that enclose string items.

5. The BYTES keyword is used to specify the maximum size of the FIELD object, not including leading or trailing delimiters or line terminators. When a field contains items, the BYTES value is set to the product of the ITEM_BYTES and ITEMS values plus the number of interior delimiter bytes (e.g., for three ASCII_INTEGER items of three bytes each ITEMS = 3, ITEM_BYTES=3, and BYTES= 11, which includes the two delimiters WITHIN the field but not the trailing delimiter).

6. The (optional) FORMAT element may be used to specify the format of FIELD data when they are present. The FORMAT specification applies to the maximum size of the field object, allowing shorter variations. For example, FORMAT = "F5.1" is consistent with each of the following:

   ... ,127.1, ...
   ... ,-12.7, ...
   ... ,3.1, ...
   ... ,3.01, ... and
   ... ,, ...

7. Inclusion of data elements VALID_MINIMUM and VALID_MAXIMUM within FIELD object definitions is encouraged.

8. If data element MISSING_CONSTANT is used, its meaning must be clearly stated since absence of a field value is the default indication of 'no data'.

A.14.1 Required Keywords

1. BYTES
2. DATA_TYPE
3. NAME

A.14.2 Optional Keywords

1. DESCRIPTION
2. FIELD_NUMBER
3. FORMAT
4. ITEM_BYTES
5. ITEMS
6. UNIT
7. VALID_MAXIMUM
8. VALID_MINIMUM
9. PSDD

A.14.3 Required Objects

None

A.14.4 Optional Objects

1. ALIAS

A.14.5 Example 1

The label fragment below shows a simple FIELD object from a SPREADSHEET object (see the SPREADSHEET section of this document).

```
OBJECT                   = FIELD
   NAME                  = "DETECTOR TEMPERATURE"
   FIELD_NUMBER          = 3
   BYTES                 = 5
   DATA_TYPE             = "ASCII_REAL"
   FORMAT                = "F5.1"
   UNIT                  = "KELVIN"
END_OBJECT               = FIELD
```

A.14.6 Example 2

The fragment below shows two FIELDs containing multiple items. The first FIELD is a vector containing three ASCII_INTEGER items: xx, yy, zz. The second FIELD contains three character items: "xx", "yy" and "zz". Note that the value of BYTES includes the comma delimiters between items, but the ITEM_BYTES value does not.

```
OBJECT                  = FIELD
  NAME                  = "FIELD 1 - IX, IY, IZ"
  DATA_TYPE             = "ASCII_INTEGER"
  FIELD_NUMBER          = 1
  BYTES                 = 8     /*includes item separating delimiters*/
  ITEMS                 = 3     /* i.e. 17,15,27   or    1,2,3       */
  ITEM_BYTES            = 2     /* individual item maximum size in bytes */
  FORMAT                = "I2"
  MISSING_CONSTANT      = -1
  DESCRIPTION           = "Raw values of FIELD 1.  IX, IY, and IZ represent
                           independent, non-negative measurements.  A value
                           of  -1  denotes a measurement that could not be
                           processed."
END_OBJECT              = FIELD

OBJECT                  = FIELD
  NAME                  = "FIELD 2 - AX, AY,AZ"
  DATA_TYPE             = "CHARACTER"
  FIELD_NUMBER          = 2     /* One FIELD object precedes this object
*/
  BYTES                 = 12    /* Doesn't include first/last quotes  */
  ITEMS                 = 3     /* i.e. "xx","yy","zz"  */
  ITEM_BYTES            = 2
  FORMAT                = "A2"
END_OBJECT              = FIELD
```

A.15 FILE

The FILE object is used in attached or detached labels to define the attributes or characteristics of a data file. In attached labels, the file object is also used to indicate boundaries between label records and data records in data files which have attached labels. The FILE object may be used in three ways:

1. As an implicit object in attached or detached labels. All detached label files and attached labels contain an implicit FILE object which starts at the top of the label and ends where the label ends. In these cases, the PDS recommends against using the NAME keyword to reference the file name. This label fragment shows the required FILE object elements as they typically appear in labels:

```
RECORD_TYPE        = FIXED_LENGTH
RECORD_BYTES       = 80
FILE_RECORDS       = 522
LABEL_RECORDS      = 10
```

   For data products labelled using the implicit file object (e.g., in minimal labels) “DATA_OBJECT_TYPE = FILE” should be used in the DATA_SET catalog object.

2. As an explicit object which is used when a file reference is needed in a combined detached or minimal label. In this case, the optional FILE_NAME element is used to identify the file being referenced.

```
OBJECT             = FILE
  FILE_NAME        = "IM10347.DAT"
  RECORD_TYPE      = STREAM
  FILE_RECORDS     = 1024
   ...
END_OBJECT         = FILE
```

   For data products labelled using the explicit FILE object (e.g., in minimal labels) DATA_OBJECT_TYPE = FILE should be used in the DATA_SET catalog object.

3. As an explicit object to identify specific files as sub-objects of the DIRECTORY in VOLUME objects. In this case, the optional FILE_NAME element is used to identify the file being referenced on a tape archive volume.

```
OBJECT             = FILE
  FILE_NAME        = "VOLDESC.CAT"
  RECORD_TYPE      = STREAM
  SEQUENCE_NUMBER  = 1
END_OBJECT         = FILE
```

The keywords in the FILE object always describe the file being referenced, and not the file in which the keywords are contained (i.e., if the FILE object is used in a detached label file, the FILE object keywords describe the detached data file, not the label file which contains the keywords). For example, if a detached label for a data file is being created and the label will be in STREAM format, but the data will be stored in a file having FIXED_LENGTH records, then the RECORD_TYPE keyword in the label file must be given the value FIXED_LENGTH.

The following table identifies data elements that are required (Req), optional (Opt), and not applicable (-) for various types of files

Labeling Method	**Att**	**Det**	**Att**	**Det**	**Att**	**Det**	**Att**	**Det**
RECORD_TYPE	FIXED_LENGTH		VARIABLE_LENGTH		STREAM		UNDEFINED	
RECORD_BYTES	Req	Req	Rmax	Rmax	Omax	-	-	-
FILE_RECORDS	Req	Req	Req	Req	Opt	Opt	-	-
LABEL_RECORDS	Req	-	Req	-	Opt	-	-	-

A.15.1 Required Keywords

1. RECORD_TYPE

(See above table for the conditions of use of additional required keywords)

A.15.2 Optional Keywords

1. DESCRIPTION
2. ENCODING_TYPE
3. FILE_NAME (required only in minimal detached labels and tape archives)
4. FILE_RECORDS (required only in minimal detached labels and tape archives)
5. INTERCHANGE_FORMAT
6. LABEL_RECORDS
7. RECORD_BYTES
8. REQUIRED_STORAGE_BYTES
9. SEQUENCE_NUMBER
10. UNCOMPRESSED_FILE_NAME

A.15.3 Required Objects

None

A.15.4 Optional Objects

None

A.15.5 Example

Following is an example of a set of explicit FILE objects in a combined detached label. An additional example of the use of explicit FILE object can be found under the VOLUME object (Section A.29).

```
PDS_VERSION_ID                 = PDS3
HARDWARE_MODEL_ID              = "SUN SPARC STATION"
OPERATING_SYSTEM_ID            = "SUN OS 4.1.1"
SPACECRAFT_NAME                = "VOYAGER 2"
INSTRUMENT_NAME                = "PLASMA WAVE RECEIVER"
MISSION_PHASE_NAME             = "URANUS ENCOUNTER"
TARGET_NAME                    = URANUS
DATA_SET_ID                    = "VG2-U-PWS-4-RDR-SA-48.0SEC-V1.0"
PRODUCT_ID                     = "T860123-T860125"

OBJECT                         = FILE
  FILE_NAME                    = "T860123.DAT"
  FILE_RECORDS                 = 1800
  RECORD_TYPE                  = FIXED_LENGTH
  RECORD_BYTES                 = 105
  START_TIME                   = 1986-01-23T00:00:00.000
  STOP_TIME                    = 1986-01-24T00:00:00.000
  ^TIME_SERIES                 = "T860123.DAT"

  OBJECT                       = TIME_SERIES
    INTERCHANGE_FORMAT         = BINARY
    ROWS                       = 1800
    ROW_BYTES                  = 105
    COLUMNS                    = 19
    ^STRUCTURE                 = "PWS_DATA.FMT"
    SAMPLING_PARAMETER_NAME    = TIME
    SAMPLING_PARAMETER_UNIT    = SECOND
    SAMPLING_PARAMETER_INTERVAL = 48.0
  END_OBJECT                   = TIME_SERIES
END_OBJECT                     = FILE

OBJECT                         = FILE
  FILE_NAME                    = "T860124.DAT"
  FILE_RECORDS                 = 1800
  RECORD_TYPE                  = FIXED_LENGTH
  RECORD_BYTES                 = 105
  START_TIME                   = 1986-01-24T00:00:00.000
  STOP_TIME                    = 1986-01-25T00:00:00.000
  ^TIME_SERIES                 = "T860124.DAT"
```

```
    OBJECT                        = TIME_SERIES
      INTERCHANGE_FORMAT          = BINARY
      ROWS                        = 1800
      ROW_BYTES                   = 105
      COLUMNS                     = 19
      ^STRUCTURE                  = "PWS_DATA.FMT"
      SAMPLING_PARAMETER_NAME     = TIME
      SAMPLING_PARAMETER_UNIT     = SECOND
      SAMPLING_PARAMETER_INTERVAL = 48.0
    END_OBJECT                    = TIME_SERIES
  END_OBJECT                      = FILE

  OBJECT                          = FILE
    FILE_NAME                     = "T860125.DAT"
    FILE_RECORDS                  = 1799
    RECORD_TYPE                   = FIXED_LENGTH
    RECORD_BYTES                  = 105
    START_TIME                    = 1986-01-30T00:00:00.000
    STOP_TIME                     = 1986-01-30T23:59:12.000
    ^TIME_SERIES                  = "T860125.DAT"

    OBJECT                        = TIME_SERIES
      INTERCHANGE_FORMAT          = BINARY
      ROWS                        = 1799
      ROW_BYTES                   = 105
      COLUMNS                     = 19
      ^STRUCTURE                  = "PWS_DATA.FMT"
      SAMPLING_PARAMETER_NAME     = TIME
      SAMPLING_PARAMETER_UNIT     = SECOND
      SAMPLING_PARAMETER_INTERVAL = 48.0
    END_OBJECT                    = TIME_SERIES
  END_OBJECT                      = FILE
  END
```

A.16 GAZETTEER_TABLE

The GAZETTEER_TABLE object is a specific type of TABLE object that provides information about the geographical features of a planet or satellite. It contains information about named features such as location, size, origin of feature name, and so on. The GAZETTEER_TABLE contains one row for each named feature on the target body. The table is formatted so that it may be read directly by many data management systems on various host computers. All fields (columns) are separated by commas, and character fields are enclosed by double quotation marks. Each record consist of 480 bytes, with a carriage return/line feed sequence in bytes 479 and 480. This allows the table to be treated as a fixed length record file on hosts that support this file type and as a normal text file on other hosts.

Currently the PDS Imaging Node at the USGS is the data producer for all GAZETTEER_TABLEs.

A.16.1 Required Keywords

1. NAME
2. INTERCHANGE_FORMAT
3. ROWS
4. COLUMNS
5. ROW_BYTES
6. DESCRIPTION

A.16.2 Optional Keywords

Any

A.16.3 Required Objects

1. COLUMN

A.16.3.1 Required COLUMN Objects (NAME =)

TARGET_NAME
SEARCH_FEATURE_NAME
DIACRITIC_FEATURE_NAME
MINIMUM_LATITUDE
MAXIMUM_LATITUDE
CENTER_LATITUDE
MINIMUM_LONGITUDE
MAXIMUM_LONGITUDE

CENTER_LONGITUDE
LABEL_POSITION_ID
FEATURE_LENGTH
PRIMARY_PARENTAGE_ID
SECONDARY_PARENTAGE_ID
MAP_SERIAL_ID
FEATURE_STATUS_TYPE
APPROVAL_DATE
FEATURE_TYPE
REFERENCE_NUMBER
MAP_CHART_ID
FEATURE_DESCRIPTION

A.16.3.2 Required Keywords (for Required COLUMN Objects)

NAME
DATA_TYPE
START_BYTE
BYTES
FORMAT
UNIT
DESCRIPTION

A.16.4 Optional Objects

None

A.16.5 Example

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 480
FILE_RECORDS                  = 1181
PRODUCT_ID                    = XYZ
TARGET_NAME                   = MARS
^GAZETTEER_TABLE              = "GAZETTER.TAB"

OBJECT                        = GAZETTEER_TABLE
  NAME                        = "PLANETARY NOMENCLATURE GAZETTEER"
  INTERCHANGE_FORMAT          = ASCII
  ROWS                        = 1181
  COLUMNS                     = 20
  ROW_BYTES                   = 480
  DESCRIPTION                 = "The gazetteer (file: GAZETTER.TAB) is
    a table of geographical features for a planet or satellite.   It
    contains information about a named feature such as location, size,
```

```
  origin of feature name, etc. The Gazetteer Table contains one row
  for each feature named on the target body.  The table is formatted
  so that it may be read directly into many data management systems on
  various host computers. All fields (columns) are separated by
  commas, and character fields are preceded by double quotation marks.
  Each record consist of 480 bytes, with a carriage return/line feed
  sequence in bytes 479 and 480. This allows the table to be treated
  as a fixed length record file on hosts that support this file type
  and as a normal text file on other hosts."

OBJECT                        = COLUMN
  NAME                        = TARGET_NAME
  DATA_TYPE                   = CHARACTER
  START_BYTE                  = 2
  BYTES                       = 20
  FORMAT                      = "A20"
  UNIT                        = "N/A"
  DESCRIPTION                 = "The planet or satellite on which the
                                 feature is located."
END_OBJECT                    = COLUMN


OBJECT                        = COLUMN
  NAME                        = SEARCH_FEATURE_NAME
  DATA_TYPE                   = CHARACTER
  START_BYTE                  = 25
  BYTES                       = 50
  FORMAT                      = "A50"
  UNIT                        = "N/A"
  DESCRIPTION                 = "The geographical feature name with all
  diacritical marks stripped off.  This name is stored in upper case
  only so that it can be used for sorting and search purposes. This
  field should not be used to designate the name of the feature
  because it does not contain the diacritical marks. Feature names not
  containing diacritical marks can often take on a completely
  different meaning and in some cases the meaning can be deeply
  offensive."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = DIACRITIC_FEATURE_NAME
  DATA_TYPE                   = CHARACTER
  START_BYTE                  = 78
  BYTES                       = 100
  FORMAT                      = "A100"
  UNIT                        = "N/A"
  DESCRIPTION                 = "The geographical feature name
  containing standard diacritical information.  A detailed description
  of the diacritical mark formats are described in  the gazetteer
  documentation.

     DIACRITICALS USED IN THE TABLE
```

The word diacritic comes from a Greek word meaning to separate. It refers to the accent marks employed to separate, or distinguish, one form of pronunciation of a vowel or consonant from another.

This note is included to familiarize the user with the codes used to represent diacriticals found in the table, and the values usually associated with them. In the table, the code for a diacritical is preceded by a backslash and is followed, without a space, by the letter it is modifying.

This note is organized as follows: the code is listed first, followed by the name of the accent mark, if applicable, a brief description of the appearance of the diacritical and a short narrative on its usage.

acute accent; a straight diagonal line extending from upper right to lower left. The acute accent is used in most languages to lengthen a vowel; in some, such as Oscan, to denote an open vowel. The acute is also often used to indicate the stressed syllable; in some transcriptions it indicates a palatalized consonant.

diaeresis or umlaut; two dots surmounting the letter. In Romance languages and English, the diaeresis is used to indicate that consecutive vowels do not form a dipthong (see below); in modern German and Scandinavian languages, it denotes palatalization of vowels.

circumflex; a chevron or inverted 'v' shape, with the apex at the top. Used most often in modern languages to indicate lengthening of a vowel.

tilde; a curving or waving line above the letter. The tilde is a form of circumflex. The tilde is used most often in Spanish to form a palatalized n as in the word 'ano', pronounced 'anyo'. It is also used occasionally to indicate nasalized vowels.

macron; a straight line above the letter. The macron is used almost universally to lengthen a vowel.

breve; a concave semicircle or 'u' shape surmounting the letter. Originally used in Greek, the breve indicates a short vowel.

a small circle or 'o' above the letter. Frequently used in Scandinavian languages to indicate a broad 'o'.

e dipthong or ligature; transcribed as two letters in contact with each other. The dipthong is a combination of vowels that are pronounced together.

cedilla; a curved line surmounted by a vertical line, placed at the bottom of the letter. The cedilla is used in Spanish and French to denote a dental, or soft, 'c'. In the new Turkish transcription, 'c' cedilla has the value of English 'ch'. In Semitic languages, the cedilla under a consonant indicates that it is emphatic.

```
  check or inverted circumflex; a 'v' shape above the letter. This
  accent is used widely in Slavic languages to indicate a palatal
  articulation, like the consonant sounds in the English words chapter
  and shoe and the 'zh' sound in pleasure.

  a single dot above the letter. This diacritical denotes various
  things; in Lithuanian, it indicates a close long vowel. In Sanskrit,
  when used with 'n', it is a velar sound,  as in the English 'sink';
  in Irish orthography, it indicates a fricative consonant (see
  below).

  accent grave; a diagonal line (above the letter) extending from
  upper left to lower right. The grave accent is used in  French,
  Spanish and Italian to denote open vowels.

  fricative; a horizontal line through a consonant. A fricative
  consonant is characterized by a frictional rustling of the breath as
  it is emitted."

END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = MINIMUM_LATITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 180
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The minimum_latitude element specifies
  the southernmost latitude of a spatial area, such as a map, mosaic,
  bin, feature, or region."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = MAXIMUM_LATITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 188
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The maximum_latitude element
  specifies the northernmost latitude of a spatial area, such as a
  map, mosaic, bin, feature, or region."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = CENTER_LATITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 196
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The center latitude of the feature."
```

```
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = MINIMUM_LONGITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 204
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The minimum_longitude element
  specifies the easternmost latitude of a spatial area, such as  a
  map, mosaic, bin, feature, or region.  "
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = MAXIMUM_LONGITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 212
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The maximum_longitude element
  specifies the westernmost longitude of a spatial area, such  as a
  map, mosaic, bin, feature, or region. "
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = CENTER_LONGITUDE
  DATA_TYPE                   = REAL
  START_BYTE                  = 220
  BYTES                       = 7
  FORMAT                      = "F7.2"
  UNIT                        = DEGREE
  DESCRIPTION                 = "The center longitude of the feature."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = LABEL_POSITION_ID
  DATA_TYPE                   = CHARACTER
  START_BYTE                  = 229
  BYTES                       = 2
  FORMAT                      = "A2"
  UNIT                        = "N/A"
  DESCRIPTION                 = "The suggested plotting position of the
  feature name (UL=Upper left, UC=Upper center, UR=Upper right,
  CL=Center left, CR=Center right, LL=Lower left,  LC=Lower center,
  LR=Lower right). This field is used to instruct the plotter where to
  place the typographical label with respect to the center of the
  feature.  This code is used to avoid crowding of names in areas
  where there is a high density of named features."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = FEATURE_LENGTH
```

```
    DATA_TYPE                  = REAL
    START_BYTE                 = 233
    BYTES                      = 8
    FORMAT                     = "F8.2"
    UNIT                       = KILOMETER
    DESCRIPTION                = "The longer or longest dimension  of an
   object. For the Gazetteer usage, this field refers to the length of
   the named feature."
  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = PRIMARY_PARENTAGE_ID
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 243
    BYTES                      = 2
    FORMAT                     = "A2"
    UNIT                       = "N/A"
    DESCRIPTION                = "This field contains the primary
   origin of the feature name (i.e. where the name originated).   It
   contains a code for the continent or country origin of the name.
   Please see Appendix 5 of the gazetteer documentation  (GAZETTER.TXT)
   for a definition of the codes used to define the continent or
   country."
  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = SECONDARY_PARENTAGE_ID
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 248
    BYTES                      = 2
    FORMAT                     = "A2"
    UNIT                       = "N/A"
    DESCRIPTION                = "This field contains the secondary
   origin of the feature name. It contains a code for a country, state,
   territory, or ethnic group. Please see  Appendix 5 of the gazetteer
   documentation (GAZETTER.TXT) for  a defintion of the codes in this
   field."

  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = MAP_SERIAL_ID
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 253
    BYTES                      = 6
    FORMAT                     = "A6"
    UNIT                       = "N/A"
    DESCRIPTION                = "The identification of the map that
   contains the named feature. This field represents the map serial
   number of the map publication used for ordering maps from the U.S.
   Geological Survey. The map identified in this field best portrays
   the named feature."
  END_OBJECT                   = COLUMN
```

```
OBJECT                          = COLUMN
  NAME                          = FEATURE_STATUS_TYPE
  DATA_TYPE                     = CHARACTER
  START_BYTE                    = 262
  BYTES                         = 12
  FORMAT                        = "A12"
  UNIT                          = "N/A"
  DESCRIPTION                   = "The IAU approval status of the named
  feature. Permitted values are 'PROPOSED', 'PROVISIONAL', 'IAU-
  APPROVED', and 'DROPPED'. Dropped names have been disallowed by the
  IAU. However, these features have been included in the gazetteer for
  historical purposes. Some named features that are disallowed by the
  IAU may commonly be used on some maps."
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
  NAME                          = APPROVAL_DATE
  DATA_TYPE                     = INTEGER
  START_BYTE                    = 276
  BYTES                         = 4
  FORMAT                        = "I4"
  UNIT                          = "N/A"
  DESCRIPTION                   = "Date at which an object has been
  approved by the officially sanctioned organization. This field
  contains the year the IAU approved the feature name."
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
  NAME                          = FEATURE_TYPE
  DATA_TYPE                     = CHARACTER
  START_BYTE                    = 282
  BYTES                         = 20
  FORMAT                        = "A20"
  UNIT                          = "N/A"
  DESCRIPTION                   = "The feature type identifies the type
  of a particular feature, according to IAU standards.  Examples are
  'CRATER', 'TESSERA', 'TERRA', etc. See Appendix 7 of the gazetteer
  documentation (GAZETTER.TXT).

  DESCRIPTOR TERMS (FEATURE TYPES)

  FEATURE                       DESCRIPTION
  ALBEDO FEATURE                Albedo feature
  CATENA                        Chain of craters
  CAVUS                         Hollows, irregular depressions
  CHAOS                         Distinctive area of broken terrain
  CHASMA                        Canyon
  COLLES                        Small hill or knob
  CORONA                        Ovoid-shaped feature
  CRATER                        Crater
  DORSUM                        Ridge
  ERUPTIVE CENTER               Eruptive center
  FACULA                        Bright spot
  FLEXUS                        Cuspate linear feature
```

```
  FLUCTUS                     Flow terrain
  FOSSA                       Long, narrow, shallow depression
  LABES                       Landslide
  LABYRINTHUS                 Intersecting valley complex
  LACUS                       Lake
  LARGE RINGED FEATURE        Large ringed feature
  LINEA                       Elongate marking
  MACULA                      Dark spot
  MARE                        Sea
  MENSA                       Mesa, flat-topped elevation
  MONS                        Mountain
  OCEANUS                     Ocean
  PALUS                       Swamp
  PATERA                      Shallow crater; scalloped, complex edge
  PLANITIA                    Low plain
  PLANUM                      Plateau or high plain
  PROMONTORIUM                Cape
  REGIO                       Region
  RIMA                        Fissure
  RUPES                       Scarp
  SCOPULUS                    Lobate or irregular scarp
  SINUS                       Bay
  SULCUS                      Subparallel furrows and ridges
  TERRA                       Extensive land mass
  TESSERA                     Tile; polygonal ground
  THOLUS                      Small domical mountain or hill
  UNDAE                       Dunes
  VALLIS                      Sinuous valley
  VASTITAS                    Widespread lowlands
  VARIABLE FEATURE            Variable feature "
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = REFERENCE_NUMBER
  DATA_TYPE                   = INTEGER
  START_BYTE                  = 304
  BYTES                       = 4
  FORMAT                      = "I4"
  UNIT                        = "N/A"
  DESCRIPTION                 = "Literature reference from which the
  spelling and description of the feature name was derived.  See
  Appendix 6 of the gazetteer documentation  (GAZETTER.TXT)."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
  NAME                        = MAP_CHART_ID
  DATA_TYPE                   = CHARACTER
  START_BYTE                  = 310
  BYTES                       = 6
  FORMAT                      = "A6"
  UNIT                        = "N/A"
  DESCRIPTION                 = "This field contains the abbreviation
  of the map designator or chart identification (example MC-19, MC-18,
  etc.)."
```

```
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = FEATURE_DESCRIPTION
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 319
    BYTES                       = 159
    FORMAT                      = "A159"
    UNIT                        = "N/A"
    DESCRIPTION                 = "Short description of the feature
                                   name."
  END_OBJECT                    = COLUMN
END_OBJECT                      = GAZETTEER_TABLE
END
```

A.17 HEADER

The HEADER object is used to identify and define the attributes of commonly used header data structures such as VICAR or FITS. These structures are usually system or software specific and are described in detail in a referenced description text file. The use of BYTES within the header object refers to the number of bytes for the entire header, not a single record.

A.17.1 Required Keywords

1. BYTES
2. HEADER_TYPE

A.17.2 Optional Keywords

1. DESCRIPTION
2. INTERCHANGE_FORMAT
3. RECORDS

A.17.3 Required Objects

None

A.17.4 Optional Objects

None

A.17.5 Example

The following example shows the detached label file “TIMTC02A.LBL”. The label describes the data product file “TIMTC02A.IMG” which contains a HEADER object followed by an IMAGE object.

```
PDS_VERSION_ID                = PDS3

/* PDS label for a TIMS image */

RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 638
FILE_RECORDS                  = 39277

/* Pointers to objects */
```

```
^IMAGE_HEADER                  = ("TIMTC02A.IMG",1)
^IMAGE                         = ("TIMTC02A.IMG",2)

/* Image description */

DATA_SET_ID                    = "C130-E-TIMS-2-EDR-IMAGE-V1.0"
PRODUCT_ID                     = "TIMTC02A"
INSTRUMENT_HOST_NAME           = "NASA C-130 AIRCRAFT"
INSTRUMENT_NAME                = "THERMAL INFRARED MULTISPECTRAL
                                  SCANNER"
TARGET_NAME                    = EARTH
FEATURE_NAME                   = "TRAIL CANYON FAN"
START_TIME                     = 1989-09-29T21:47:35
STOP_TIME                      = 1989-09-29T21:47:35
CENTER_LATITUDE                = 36.38
CENTER_LONGITUDE               = 116.96
INCIDENCE_ANGLE                = 0.0
EMISSION_ANGLE                 = 0.0

/* Description of objects */

OBJECT                         = IMAGE_HEADER
  BYTES                        = 638
  RECORDS                      = 1
  HEADER_TYPE                  = VICAR2
  INTERCHANGE_FORMAT           = BINARY
  ^DESCRIPTION                 = "VICAR2.TXT"
END_OBJECT                     = IMAGE_HEADER

OBJECT                         = IMAGE
  LINES                        = 6546
  LINE_SAMPLES                 = 638
  SAMPLE_TYPE                  = UNSIGNED_INTEGER
  SAMPLE_BITS                  = 8
  SAMPLE_BIT_MASK              = 2#11111111#
  BANDS                        = 6
  BAND_STORAGE_TYPE            = LINE_INTERLEAVED
END_OBJECT                     = IMAGE
END
```

A.18 HISTOGRAM

The HISTOGRAM object is a sequence of numeric values that provides the number of occurrences of a data value or a range of data values in a data object. The number of items in a histogram will normally be equal to the number of distinct values allowed in a field of the data object. For example, an 8-bit integer field can have a maximum of 256 values, and would result in a 256 item histogram. HISTOGRAMs may be used to bin data, in which case an offset and scaling factor indicate the dynamic range of the data represented.

The following equation allows the calculation of the range of each bin in the histogram:

bin_lower_boundary = bin_element * SCALING_FACTOR + OFFSET

A.18.1 Required Keywords

1. ITEMS
2. DATA_TYPE
3. ITEM_BYTES

A.18.2 Optional Keywords

1. BYTES
2. INTERCHANGE_FORMAT
3. OFFSET
4. SCALING_FACTOR

A.18.3 Required Objects

None

A.18.4 Optional Objects

None

A.18.5 Example

```
PDS_VERSION_ID                  = PDS3

/*          FILE FORMAT AND LENGTH */

RECORD_TYPE                     = FIXED_LENGTH
```

```
RECORD_BYTES                   = 956
FILE_RECORDS                   = 965
LABEL_RECORDS                  = 3

/*           POINTERS TO START RECORDS OF OBJECTS IN FILE */

^IMAGE_HISTOGRAM               = 4
^IMAGE                         = 6

/*           IMAGE DESCRIPTION */

DATA_SET_ID                    = "VO1/VO2-M-VIS-5-DIM-V1.0"
PRODUCT_ID                     = "MG15N022-GRN-666A"
SPACECRAFT_NAME                = VIKING_ORBITER_1
TARGET_NAME                    = MARS
START_TIME                     = 1978-01-14T02:00:00
STOP_TIME                      = 1978-01-14T02:00:00
SPACECRAFT_CLOCK_START_TIME    = UNK
SPACECRAFT_CLOCK_STOP_TIME     = UNK
PRODUCT_CREATION_TIME          = 1995-01-01T00:00:00
ORBIT_NUMBER                   = 666
FILTER_NAME                    = GREEN
IMAGE_ID                       = "MG15N022-GRN-666A"
INSTRUMENT_NAME                = {VISUAL_IMAGING_SUBSYSTEM_CAMERA_A,
                                  VISUAL_IMAGING_SUBSYSTEM_CAMERA_B}
NOTE                           = "MARS MULTI-SPECTRAL MDIM SERIES"

/* SUN RAYS EMISSION, INCIDENCE, AND PHASE ANGLES OF IMAGE CENTER*/

SOURCE_PRODUCT_ID              = "666A36"
EMISSION_ANGLE                 = 21.794
INCIDENCE_ANGLE                = 66.443
PHASE_ANGLE                    = 46.111

/*           DESCRIPTION OF OBJECTS CONTAINED IN FILE */

OBJECT                         = IMAGE_HISTOGRAM
  ITEMS                        = 256
  DATA_TYPE                    = VAX_INTEGER
  ITEM_BYTES                   = 4
END_OBJECT                     = IMAGE_HISTOGRAM

OBJECT                         = IMAGE
  LINES                        = 960
  LINE_SAMPLES                 = 956
  SAMPLE_TYPE                  = UNSIGNED_INTEGER
  SAMPLE_BITS                  = 8
  SAMPLE_BIT_MASK              = 2#11111111#
  CHECKSUM                     = 65718982

  /*  I/F = SCALING_FACTOR*DN + OFFSET, CONVERT TO INTENSITY/FLUX */

  SCALING_FACTOR               = 0.001000
  OFFSET                       = 0.0

  /* OPTIMUM COLOR STRETCH FOR DISPLAY OF COLOR IMAGES */

  STRETCHED_FLAG               = FALSE
```

```
  STRETCH_MINIMUM             = ( 53,  0)
  STRETCH_MAXIMUM             = (133,255)
END_OBJECT                    = IMAGE

END
```

A.19 HISTORY

A HISTORY object is a dynamic description of the history of one or more associated data objects in a file. It supplements the essentially static description contained in the PDS label.

The HISTORY object contains text in a format similar to that of the ODL statements used in the label. It identifies previous computer manipulation of the principal data object(s) in the file. It includes an identification of the source data, processes performed, processing parameters, as well as dates and times of processing. It is intended that the history be available for display, be dynamically extended by any process operating on the data, and be automatically propagated to the resulting data file. Eventually, it might be extracted for loading in detailed level catalogs of data set contents.

The HISTORY object is structured as a series of History Entries, one for each process which has operated on the data. Each entry contains a standard set of ODL element assignment statements, delimited by "GROUP = *program_name*" and "END_GROUP = *program_name*" statements. A subgroup in each entry, delimited by "GROUP = PARAMETERS" and "END_GROUP = PARAMETERS", contains statements specifying the values of all parameters of the program.

A.19.1 HISTORY ENTRY ELEMENTS

Attribute	**Description**
VERSION_DATE	Program version date, ISO standard format.
DATE_TIME	Run date and time, ISO standard format.
NODE_NAME	Network name of computer.
USER_NAME	Username.
SOFTWARE_DESC	Program-generated (brief) description.
USER_NOTE	User-supplied (brief) description.

Unlike the above elements, the names of the parameters defined in the PARAMETERS subgroup are uncontrolled, and must only conform to the program.

The last entry in a HISTORY object is followed by an END statement. The HISTORY object, by convention, follows the PDS label of the file, beginning on a record boundary, and is located by a pointer statement in the label. There are no required elements for the PDS label description of the object; it is represented in the label only by the pointer statement, and OBJECT = HISTORY and END_OBJECT = HISTORY statements.

The HISTORY capability has been implemented as part of the Integrated Software for Imaging Spectrometers (ISIS) system (see QUBE object definition). ISIS QUBE applications add their own entries to the QUBE file's cumulative HISTORY object. ISIS programs run under NASA's TAE (Transportable Applications Executive) system, and are able to automatically insert all parameters of their TAE procedure into the HISTORY entry created by the program. Consult the

ISIS System Design document for details and limitations imposed by that system. (See the QUBE object description for further references.)

A.19.2 Required Keywords

None

A.19.3 Optional Keywords

None

A.19.4 Required Objects

None

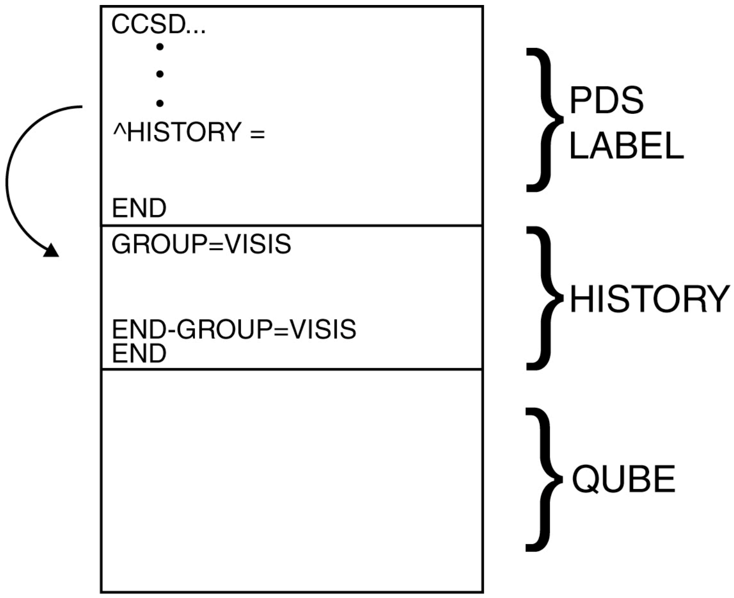
A.19.5 Optional Objects

None

A.19.6 Example

The following single-entry HISTORY object is from a Vicar-generated PDS-labeled QUBE file. (See the QUBE object example.) There is only one entry because the QUBE (or rather its label) was generated by a single program, VISIS. A QUBE generated by multiple ISIS programs would have multiple history entries, represented by multiple GROUPs in the HISTORY object.

The diagram following illustrates the placement of the example HISTORY object within a QUBE data product with an attached PDS label.




```
OBJECT                              = HISTORY
GROUP                               = VISIS

  VERSION_DATE                      = 1990-11-08
  DATE_TIME                         = 1991-07-25T10:12:52
  SOFTWARE_DESC                     = "ISIS cube file with PDS label has
     been generated as systematic product by MIPL using the following
     programs:

     NIMSMERGE to create EDR's;
     NIMSCMM to create the merged mosaic & geometry cube;
     HIST2D to create a two-dimensional histogram;
     SPECPLOT to create the spectral plots;
     TRAN, F2, and INSERT3D to create the SII cube;
     VISIS to create the ISIS cube."
  USER_NOTE                         = "VPDIN1/ Footprint, Limbfit,
                                   Height=50"

  GROUP                             = PARAMETERS
    EDR_FILE_NAME                   = " "
                                 /*EDR accessed through MIPL Catalog*/
    IMAGE_ID                        = NULL
    SPICE_FILE_NAME                 = "N/A"
    SPIKE_FILE_NAME                 =
                                   "MIPL:[MIPL.GLL]BOOM_OBSCURATION.NIM"
    DARK_VALUE_FILE_NAME            = "N/A"
    CALIBRATION_FILE_NAME           = "NDAT:NIMSGS2.CAL"
    MERGED_MOSAIC_FILE_NAME         = "NDAT:VPDIN1_DN_FP_LF_H50.CUB"
    DARK_INTERPOLATION_TYPE         = NOUPDAT
    PHOTOMETRIC_CORRECTION_TYPE = NONE
    CUBE_NIMSEL_TYPE                = NOCAL
    BINNING_TYPE                    = FOOTPRNT
    FILL_BOX_SIZE                   = 0
    FILL_MIN_VALID_PIXELS           = 0
```

```
    SUMMARY_IMAGE_RED_ID          = 0
    SUMMARY_IMAGE_GREEN_ID        = 0
    SUMMARY_IMAGE_BLUE_ID         = 0
    ADAPT_STRETCH_SAT_FRAC        = 0.000000
    ADAPT_STRETCH_SAMP_FRAC       = 0.000000
    RED_STRETCH_RANGE             = (    0,     0)
    GREEN_STRETCH_RANGE           = (    0,     0)
    BLUE_STRETCH_RANGE            = (    0,     0)
  END_GROUP                       = PARAMETERS
END_GROUP                         = VISIS
END_OBJECT                        = HISTORY
END
```

A.20 IMAGE

An IMAGE object is a two-dimensional array of values, all of the same type, each of which is referred to as a *sample*. IMAGE objects are normally processed with special display tools to produce a visual representation of the samples by assigning brightness levels or display colors to the values. An IMAGE consists of a series of lines, each containing the same number of samples.

The required IMAGE keywords define the parameters for simple IMAGE objects:

- LINES is the number of lines in the image.
- LINE_SAMPLES is the number of samples in each line.
- SAMPLE_BITS is the number of bits in each individual sample.
- SAMPLE_TYPE defines the sample data type.

In more complex images, each individual line may have some attached data which are not part of the image itself (engineering data, checksums, time tags, etc.). In this case the additional, non-image parameters are accounted for as either LINE_PREFIX_BYTES or LINE_SUFFIX_BYTES, depending on whether they occur before or after the image samples in the line. These keywords indicate the total number of bytes used for the additional data, so that software processing the image can clip these bytes before attempting to display or manipulate the image. The structure of the prefix or suffix bytes is most often defined by a TABLE object (in the same label), which will itself have ROW_SUFFIX_BYTES or ROW_PREFIX_BYTES, to allow table-processing software to skip over the image data. Figure A.1 illustrates the layout of prefix and suffix bytes around an image.

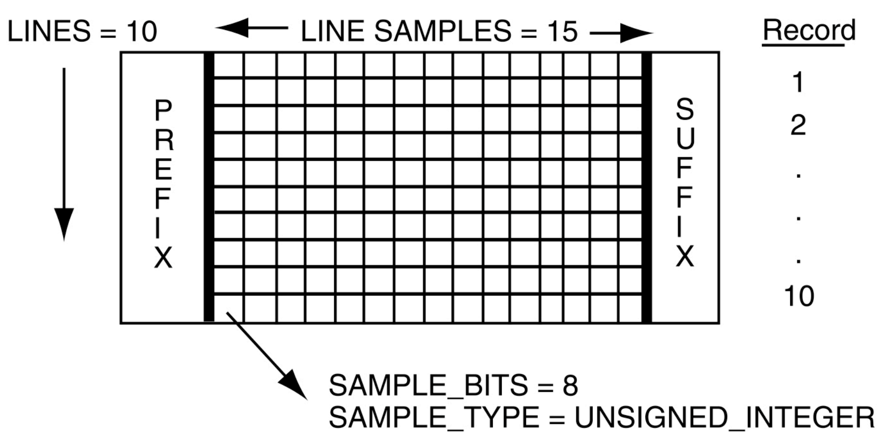


Figure A.1 – Prefix and Suffix Bytes Attached to an Image

Sometimes a single image is composed of several bands of data. For example, a color image for video display may actually consist of three copies of the image: one in red, one in green and one in blue. Each logical sample corresponds to one value for each of the bands. In this case, the keyword BANDS is used to indicate the presence of multiple bands of data. BAND_STORAGE_TYPE indicates how the banded values are organized:

- SAMPLE_INTERLEAVED means that in each line, all band values for each sample are adjacent in the line. So in the above example of an RGB image, each line would look like this (numbers are sample numbers, RGB = red, green, blue):

```
1R 1G 1B  2R 2G 2B  3R 3G 3B ...
```

- LINE_INTERLEAVED means that successive lines contain the band values for corresponding samples. Continuing with the RGB example, the first physical lines in the image data would represent the first display line of the image, first in red, then green, then blue:

```
1R 2R 3R 4R ...
1G 2G 3G 4G ...
1B 2B 3B 4B ...
```

By default, IMAGE objects should be displayed so that the samples are drawn from left to right and the lines from top to bottom. Other organizations can be indicated by using the LINE_DISPLAY_DIRECTION and SAMPLE_DISPLAY_DIRECTION keywords. Note that when using one of these keywords in a label, the other keyword is also required. The keywords must have orthogonal values.

Figure A.2 illustrates band storage schemes and the related keyword values.

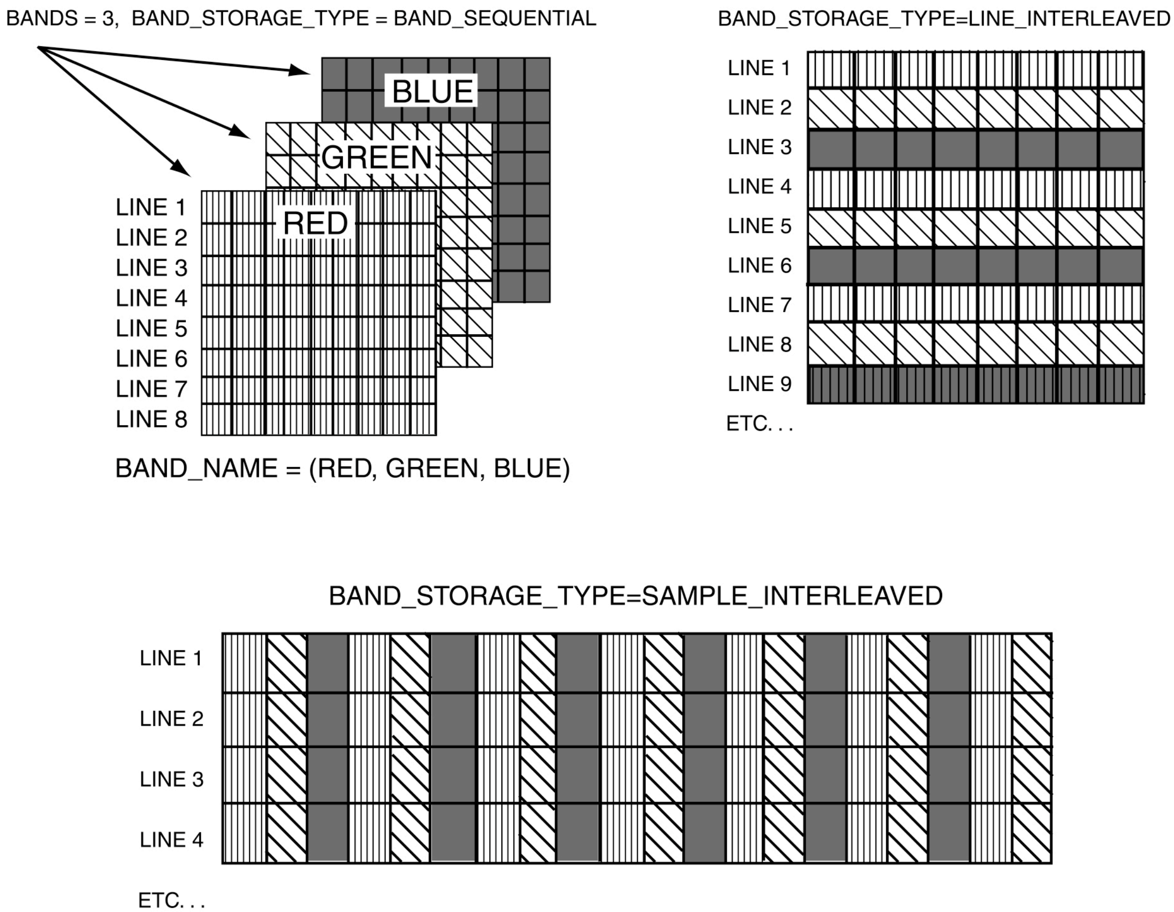


Figure A.2 – Keywords for a Multi-Band Image

A.20.1 Required Keywords

1. LINES
2. LINE_SAMPLES
3. SAMPLE_TYPE
4. SAMPLE_BITS

A.20.2 Optional Keywords

1. BAND_SEQUENCE
2. BAND_STORAGE_TYPE
3. BANDS
4. CHECKSUM
5. DERIVED_MAXIMUM

6. DERIVED_MINIMUM
7. DESCRIPTION
8. ENCODING_TYPE
9. FIRST_LINE
10. FIRST_LINE_SAMPLE
11. INVALID_CONSTANT
12. LINE_DISPLAY_DIRECTION
13. LINE_PREFIX_BYTES
14. LINE_SUFFIX_BYTES
15. MISSING _CONSTANT
16. OFFSET
17. SAMPLE_BIT_MASK
18. SAMPLE_DISPLAY_DIRECTION
19. SAMPLING_FACTOR
20. SCALING_FACTOR
21. SOURCE_FILE_NAME
22. SOURCE_LINES
23. SOURCE_LINE_SAMPLES
24. SOURCE_SAMPLE_BITS
25. STRETCHED_FLAG
26. STRETCH_MINIMUM
27. STRETCH_MAXIMUM

A.20.3 Required Objects

None

A.20.4 Optional Objects

1. WINDOW

A.20.5 Example

This is an example of an (attached) IMAGE label for a color digital mosaic image from the Mars Digital Image Map CD-ROMs. It includes a CHECKSUM to support automated volume production and validation, a SCALING_FACTOR to indicate the relationship between sample values and geophysical parameters and stretch keywords to indicate optimal values for image display.

```
PDS_VERSION_ID                  = PDS3

RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 956
FILE_RECORDS                    = 965
LABEL_RECORDS                   = 3
```

```
^IMAGE_HISTOGRAM              = 4
^IMAGE                        = 6

DATA_SET_ID                   = "VO1/VO2-M-VIS-5-DIM-V1.0"
PRODUCT_ID                    = "MG15N022-GRN-666A"
SPACECRAFT_NAME               = VIKING_ORBITER_1
TARGET_NAME                   = MARS
IMAGE_TIME                    = 1978-01-14T02:00:00
START_TIME                    = UNK
STOP_TIME                     = UNK
SPACECRAFT_CLOCK_START_COUNT  = UNK
SPACECRAFT_CLOCK_STOP_COUNT   = UNK
PRODUCT_CREATION_TIME         = 1995-01-01T00:00:00
ORBIT_NUMBER                  = 666
FILTER_NAME                   = GREEN
IMAGE_ID                      = "MG15N022-GRN-666A"
INSTRUMENT_NAME               = {VISUAL_IMAGING_SUBSYSTEM_CAMERA_A,
                                 VISUAL_IMAGING_SUBSYSTEM_CAMERA_B}
NOTE                          = "MARS MULTI-SPECTRAL MDIM SERIES"
SOURCE_PRODUCT_ID             = "666A36"
EMISSION_ANGLE                = 21.794
INCIDENCE_ANGLE               = 66.443
PHASE_ANGLE                   = 46.111

/*  DESCRIPTION OF OBJECTS CONTAINED IN FILE */

OBJECT                        = IMAGE_HISTOGRAM
  ITEMS                       = 256
  DATA_TYPE                   = VAX_INTEGER
  ITEM_BYTES                  = 4
END_OBJECT                    = IMAGE_HISTOGRAM

OBJECT                        = IMAGE
  LINES                       = 960
  LINE_SAMPLES                = 956
  SAMPLE_TYPE                 = UNSIGNED_INTEGER
  SAMPLE_BITS                 = 8
  SAMPLE_BIT_MASK             = 2#11111111#
  CHECKSUM                    = 65718982
  SCALING_FACTOR              = 0.001000
                                /* I/F = scaling factor*DN+offset, */
                                /* convert to intensity/flux.      */
  OFFSET                      = 0.0
  STRETCHED_FLAG              = FALSE
                                /* Optimum color stretch for display
                                */
                                /* of color images.               */
  STRETCH_MINIMUM             = ( 53,  0)
  STRETCH_MAXIMUM             = (133,255)
END_OBJECT                    = IMAGE

END
```

A.21 INDEX_TABLE

The INDEX_TABLE object is a specific type of a TABLE object that provides information about the data stored on an archive volume. The INDEX_TABLE contains one row for each data file (or data product label file, in the case where detached labels are used) on the volume. The table is formatted so that it may be read directly by many data management systems on various host computers: all fields (columns) are separated by commas; character fields are enclosed in double quotation marks; and each record ends in a carriage return/line feed sequence.

The columns of an INDEX_TABLE contain path information for each file, plus values extracted from keywords in the PDS labels. Columns are selected to allow users to a) search the table for specific files of interest; and b) identify the exact location of the file both on the volume and in the PDS catalog. In general, the columns listed in Section A.20.5.1 as *optional* are used for searching the table; the *required* columns listed in Section A.20.4.1 provide the identification information for each file. Where possible the PDS keyword name should be used as the NAME value in the corresponding COLUMN definition.

Note: See Section 17.2 for information about the use of the constants "N/A", "UNK" and "NULL" in an INDEX_TABLE.

A.21.1 INDEX_TABLEs Under Previous Version of the Standards

Prior to version 3.2 of the Standards, the INDEX_TYPE keyword was optional. Cumulative indices were identified by their filenames, which were (and still are) of the form "CUMINDEX.TAB" or "*axx*CMIDX.TAB" (with *axx* representing up to three alphanumeric characters). So, when INDEX_TYPE is not present, it defaults to "CUMULATIVE" in cumulative index files (that is, file with filenames as above) and "SINGLE" in all other index files.

A.21.2 Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES
5. INDEX_TYPE

A.21.3 Optional Keywords

1. NAME
2. DESCRIPTION
3. INDEXED_FILE_NAME

4. UNKNOWN_CONSTANT
5. NOT_APPLICABLE_CONSTANT

A.21.4 Required Objects

1. COLUMN

A.21.4.1 Required COLUMN Objects

The following COLUMN objects (as identified by the COLUMN_NAME keyword) are required to be included in the INDEX_TABLE object:

COLUMN_NAME
1. FILE_SPECIFICATION_NAME, or PATH_NAME and FILE_NAME
2. PRODUCT_ID **
3. VOLUME_ID *
4. DATA_SET_ID *
5. PRODUCT_CREATION_TIME *
6. LOGICAL_VOLUME_PATH_NAME * (must be used with PATH_NAME and FILE_NAME for a logical volume)

* If the value is constant across the data in the index table, this keyword can appear in the index table's label. If the value is not constant, then a column of the given name must be used.

** PRODUCT_ID is not required if it has the same value as FILE_NAME or FILE_SPECIFICATION_NAME.

A.21.4.2 Required Keywords (for Required COLUMN Objects)

1. NAME
2. DATA_TYPE
3. START_BYTE
4. BYTES
5. DESCRIPTION

A.21.5 Optional Objects

None

A.21.5.1 Optional COLUMN Objects (NAME=)

The following COLUMN objects (as identified by the COLUMN_NAME keyword) may be optionally included in the INDEX_TABLE object:

COLUMN_NAME

1. MISSION_NAME
2. INSTRUMENT_NAME (or ID)
3. INSTRUMENT_HOST_NAME (or ID), or SPACECRAFT_NAME (or ID)
4. TARGET_NAME
5. PRODUCT_TYPE
6. MISSION_PHASE_NAME
7. VOLUME_SET_ID
8. START_TIME
9. STOP_TIME
10. SPACECRAFT_CLOCK_START_COUNT
11. SPACECRAFT_CLOCK_STOP_COUNT
12. any other search columns

A.21.6 Example

```
PDS_VERSION_ID                = PDS3

 RECORD_TYPE                  = FIXED_LENGTH
 RECORD_BYTES                 = 180
 FILE_RECORDS                 = 220
 DESCRIPTION                  = "INDEX.TAB lists all data files on this
                                 volume"
 ^INDEX_TABLE                 = "INDEX.TAB"

 OBJECT                       = INDEX_TABLE
   INTERCHANGE_FORMAT         = ASCII
   ROW_BYTES                  = 180
   ROWS                       = 220
   COLUMNS                    = 9
   INDEX_TYPE                 = SINGLE
   INDEXED_FILE_NAME          = {"*.AMD","*.ION","*.TIM","*.TRO",
                                 "*.WEA","*.LIT","*.MIF","*.MPD",
                                 "*.ODF","*.ODR","*.ODS","*.SFO",
                                 "*.SOE","*.TDF"}

   OBJECT                     = COLUMN
     NAME                     = VOLUME_ID
     DESCRIPTION              = "Identifies the volume containing  the
                                 named file"
     DATA_TYPE                = CHARACTER
     START_BYTE               = 2
     BYTES                    = 9
   END_OBJECT                 = COLUMN
```

```
OBJECT                         = COLUMN
  NAME                         = DATA_SET_ID
  DESCRIPTION                  = "The data set identifier. Acceptable
                                  values include 'MO-M-RSS-1-OIDR-V1.0'"
  DATA_TYPE                    = CHARACTER
  START_BYTE                   = 14
  BYTES                        = 25
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = PATH_NAME
  DESCRIPTION                  = "Path to directory containing file.
    Acceptable values include:
                                'AMD',
                                'ION',
                                'TIM',
                                'TRO',
                                'WEA',
                                'LIT',
                                'MIF',
                                'MPD',
                                'ODF',
                                'ODR',
                                'ODS',
                                'SFO',
                                'SOE', and
                                'TDF'."
  DATA_TYPE                    = CHARACTER
  START_BYTE                   = 42
  BYTES                        = 9
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = FILE_NAME
  DESCRIPTION                  = "Name of file in archive"
  DATA_TYPE                    = CHARACTER
  START_BYTE                   = 54
  BYTES                        = 12
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = PRODUCT_ID
  DESCRIPTION                  = "Original file name on MO PDB or  SOPC"
  DATA_TYPE                    = CHARACTER
  START_BYTE                   = 69
  BYTES                        = 33
END_OBJECT                     = COLUMN

OBJECT                         = COLUMN
  NAME                         = START_TIME
  DESCRIPTION                  = "Time at which data in the file begin
 given in the format 'YYYY-MM-DDThh:mm:ss'."
  DATA_TYPE                    = CHARACTER
  START_BYTE                   = 105
```

```
    BYTES                      = 19
  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = STOP_TIME
    DESCRIPTION                = "Time at which data in the file end
   given in the format 'YYYY-MM-DDThh:mm:ss'."
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 127
    BYTES                      = 19
  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = PRODUCT_CREATION_TIME
    DESCRIPTION                = "Date and time that file was created."
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 149
    BYTES                      = 19
  END_OBJECT                   = COLUMN

  OBJECT                       = COLUMN
    NAME                       = FILE_SIZE
    DESCRIPTION                = "Number of bytes in file, not
                                  including label."
    DATA_TYPE                  = "ASCII INTEGER"
    START_BYTE                 = 170
    BYTES                      = 9
  END_OBJECT                   = COLUMN

END_OBJECT                     = INDEX_TABLE
END
```

A.22 PALETTE

The PALETTE object, a sub-class of the TABLE object, contains entries which represent color table assignments for values (i.e., SAMPLEs) contained in an IMAGE.

If the PALETTE is stored in a separate file from the IMAGE object, then it should be stored in ASCII format as 256 rows, each with 4 columns. The first column contains the SAMPLE value (running from 0–255 for an 8-bit SAMPLE, for example), and the remaining three columns contain the relative amount (a value from 0 to 255) of each primary color to be assigned for that SAMPLE value.

If the PALETTE is stored in the same file as the IMAGE object, then the PALETTE should be stored in BINARY format as 256 consecutive 8-bit values for each primary color (RED, GREEN, BLUE) resulting in a 768-byte record.

A.22.1 Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. ROW_BYTES
4. COLUMNS

A.22.2 Optional Keywords

1. DESCRIPTION
2. NAME

A.22.3 Required Objects

1. COLUMN

A.22.4 Optional Objects

None

A.22.5 Example

The examples below illustrate both types of PALETTE objects (ASCII and BINARY). The first is example is a complete label for an ASCII PALETTE object:

```
PDS_VERSION_ID                 = PDS3
```

```
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 80
FILE_RECORDS                  = 256
^PALETTE                      = "PALETTE.TAB"

 /* Image Palette description  */
SPACECRAFT_NAME               = MAGELLAN
MISSION_PHASE_NAME            = PRIMARY_MISSION
TARGET_NAME                   = VENUS
PRODUCT_ID                    ="GEDR-MERC.1;2"
IMAGE_ID                      ="GEDR-MERC.1;2"
INSTRUMENT_NAME               ="RADAR SYSTEM"
PRODUCT_CREATION_TIME         = 1995-01-01T00:00:00
NOTE                          = "Palette for browse image"

/* Description of an ASCII PALETTE object */

OBJECT                        = PALETTE
  INTERCHANGE_FORMAT          = ASCII
  ROWS                        = 256
  ROW_BYTES                   = 80
  COLUMNS                     = 4

  OBJECT                      = COLUMN
    NAME                      = SAMPLE
    DESCRIPTION               = "DN value for red, green, blue
                                 intensities"
    DATA_TYPE                 = ASCII_INTEGER
    START_BYTE                = 1
    BYTES                     = 3
  END_OBJECT

  OBJECT                      = COLUMN
    NAME                      = RED
    DESCRIPTION               = "Red intensity (0 - 255)"
    DATA_TYPE                 = ASCII_INTEGER
    START_BYTE                = 6
    BYTES                     = 3
  END_OBJECT

  OBJECT                      = COLUMN
    NAME                      = GREEN
    DESCRIPTION               = "Green intensity (0 - 255)"
    DATA_TYPE                 = ASCII_INTEGER
    START_BYTE                = 11
    BYTES                     = 3
  END_OBJECT

  OBJECT                      = COLUMN
    NAME                      = BLUE
    DESCRIPTION               = "Blue intensity (0 - 255)"
    DATA_TYPE                 = ASCII_INTEGER
    START_BYTE                = 16
    BYTES                     = 3
```

```
  END_OBJECT
END_OBJECT
END
```

This label fragment illustrates the definition of a binary PALETTE object:

```
/* Description of a BINARY PALETTE object */

OBJECT                        = PALETTE
  INTERCHANGE_FORMAT          = BINARY
  ROWS                        = 1
  ROW_BYTES                   = 768
  COLUMNS                     = 3

  OBJECT                      = COLUMN
    NAME                      = RED
    DATA_TYPE                 = UNSIGNED_INTEGER
    START_BYTE                = 1
    ITEMS                     = 256
    ITEM_BYTES                = 1
  END_OBJECT                  = COLUMN

  OBJECT                      = COLUMN
    NAME                      = GREEN
    DATA_TYPE                 = UNSIGNED_INTEGER
    START_BYTE                = 257
    ITEMS                     = 256
    ITEM_BYTES                = 1
  END_OBJECT                  = COLUMN

  OBJECT                      = COLUMN
    NAME                      = BLUE
    DATA_TYPE                 = UNSIGNED_INTEGER
    START_BYTE                = 513
    ITEMS                     = 256
    ITEM_BYTES                = 1
  END_OBJECT                  = COLUMN
END_OBJECT                    = PALETTE
```

A.23 QUBE

A generalized QUBE object is a multidimensional array (called the core) of sample values in multiple dimensions. The core is homogeneous, and consists of unsigned byte, signed halfword or floating point fullword elements. QUBEs of one to three dimensions may have optional suffix areas in each axis. The suffix areas may be heterogeneous, with elements of different types, but each suffix pixel is always allocated a full word. Special values may be defined for the core and the suffix areas to designate missing values and several kinds of invalid values, such as instrument and representation saturation.

The QUBE is the principal data structure of the ISIS (Integrated Software for Imaging Spectrometers) system. A frequently used specialization of the QUBE object is the ISIS Standard Qube, which is a three-dimensional QUBE with two spatial dimensions and one spectral dimension. Its axes have the interpretations 'sample', 'line' and 'band'. Three physical storage orders are allowed: band-sequential, line_interleaved (band-interleaved-by-line) and sample_interleaved (band-interleaved-by-pixel).

An example of a Standard ISIS Qube is a spectral image qube containing data from an imaging spectrometer. Such a qube is simultaneously a set of images (at different wavelengths) of the same target area, and a set of spectra at each point of the target area. Typically, suffix areas in such a qube are confined to 'backplanes' containing geometric or quality information about individual spectra, i.e. about the set of corresponding values at the same pixel location in each band.

The following diagram illustrates the general structure of a Standard ISIS Qube. Note that this is a conceptual or "logical" view of the qube.

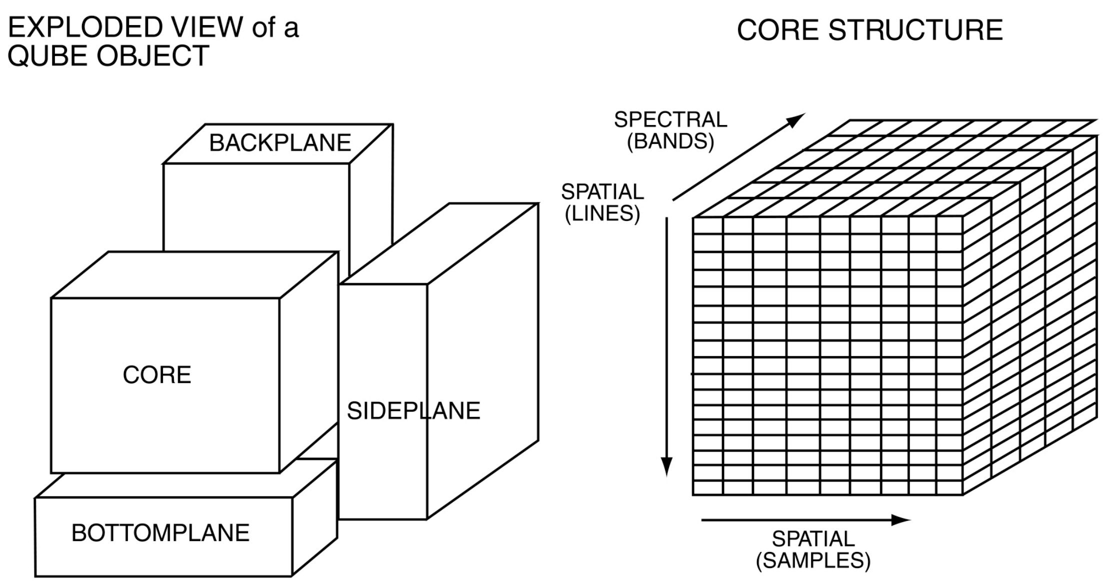


Figure A.3 – Exploded View of a Qube Object

Some special requirements are imposed by the ISIS system. A QUBE object must be associated with a HISTORY object. (Other objects, such as HISTOGRAMs, IMAGEs, PALETTEs and TABLEs which contain statistics, display parameters, engineering values or other ancillary data, are optional.) A special element, FILE_STATE, is required in the implicit FILE object. Some label information is organized into GROUPs, such as BAND_BIN and IMAGE_MAP_PROJECTION. The BAND_BIN group contains essential wavelength information, and is required for Standard ISIS Qubes.

The ISIS system includes routines for reading and writing files containing QUBE objects. Both 'logical' access, independent of actual storage order, and direct 'physical' access are provided for Standard ISIS Qubes. Only physical access is provided for generalized QUBEs. Most ISIS application programs operate on Standard ISIS Qubes. Arbitrary subqubes ('virtual' qubes) of existing qubes may be specified for most of these programs. In addition, ISIS includes software for handling Tables (an ISIS variant of the PDS Table object) and Instrument Spectral Libraries.

For a complete description, refer to the most recent version of "ISD: ISIS System Design, Build 2", obtainable from the PDS Operator.

NOTE: The following required and optional elements of the QUBE object are ISIS-specific. Since the ISIS system was designed before the current version of the Planetary Science Data Dictionary, some of the element names conflict with current PDS nomenclature standards.

A.23.1 Required Keywords (Generalized Qube and Standard ISIS Qube)

AXES	Number of axes or dimensions of qube [integer]
AXIS_NAME	Names of axes [sequence of 1-6 literals]
	(BAND, LINE, SAMPLE) for Standard Qube
CORE_ITEMS	Core dimensions of axes [seq of 1-6 integers]
CORE_ITEM_BYTES	Core element size [integer bytes: {1, 2, 4}]
CORE_ITEM_TYPE	Core element type
	[literal: {UNSIGNED_INTEGER, INTEGER, REAL}]
CORE_BASE	Base value of core item scaling [real]
CORE_MULTIPLIER	Multiplier for core item scaling [real]
	'true' value = base + multiplier * 'stored' value
	(base = 0.0 and multiplier = 1.0 for REALs)
SUFFIX_BYTES	Storage allocation of suffix elements [integer: always 4]
SUFFIX_ITEMS	Suffix dimensions of axes [seq of 1-6 integers]

CORE_VALID_MINIMUM	Minimum valid core value -- values below this value are reserved for 'special' values, of which 5 are currently assigned [integer or non-decimal integer: these values are fixed by ISIS convention for each allowable item type and size -- see ISD for details]
CORE_NULL	Special value indicating 'invalid' data
CORE_LOW_INSTR_SATURATION	Special value indicating instrument saturation at the low end
CORE_HIGH_INSTR_SATURATION	Special value indicating instrument saturation at the high end
CORE_LOW_REPR_SATURATION	Special value indicating representation saturation at the low end
CORE_HIGH_REPR_SATURATION	Special value indicating representation saturation at the high end

A.23.2 Required Keywords (Standard ISIS Qube) and Optional Keywords (Generalized Qube)

CORE_NAME	Name of value stored in core of qube [literal, e.g. SPECTRAL_RADIANCE]
CORE_UNIT	Unit of value stored in core of qube [literal]
BAND_BIN_CENTER	Wavelengths of bands in a Standard Qube [sequence of reals]
BAND_BIN_UNIT	Unit of wavelength [literal, e.g. MICROMETER]
BAND_BIN_ORIGINAL_BAND	Original band numbers, referring to a Qube of which the current qube is a subqube. In the original qube, these are sequential integers.[sequence of integers]

A.23.3 Optional Keywords (Generalized Qube and Standard ISIS Qube)

BAND_BIN_WIDTH	Width (at half height) of spectral response of bands [sequence of reals]
BAND_BIN_STANDARD_DEVIATION	Standard deviation of spectrometer values at each band [sequence of reals]
BAND_BIN_DETECTOR	Instrument detector number of band, where relevant [sequence of integers]

BAND_BIN_GRATING_POSITION	Instrument grating position of band, where relevant [sequence of integers]

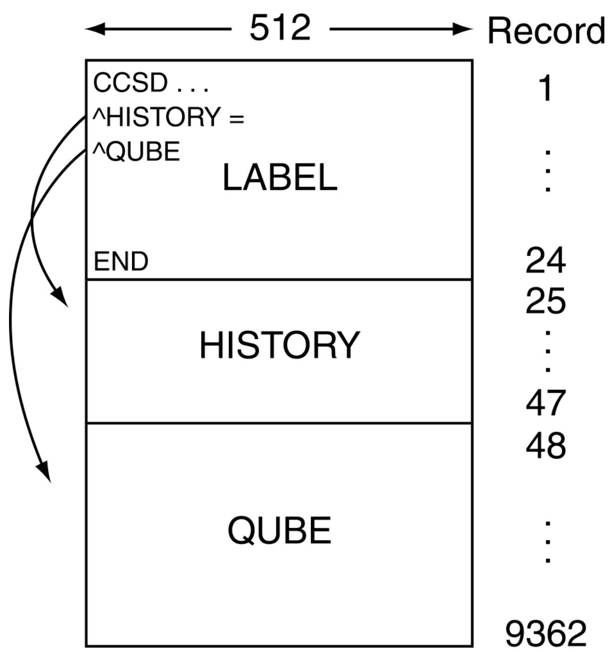
A.23.3.1 **Required Keywords (for each suffix present in a 1-3 dimensional** qube):

Note: These must be prefixed by the specific AXIS_NAME. These are SAMPLE, LINE and BAND for Standard ISIS Qubes. Only the commonly used BAND variants are shown:

BAND_SUFFIX_NAME	Names of suffix items [sequence of literals]
BAND_SUFFIX_UNIT	Units of suffix items [sequence of literals]
BAND_SUFFIX_ITEM_BYTES	Suffix item sizes [sequence of integer bytes {1, 2, 4}]
BAND_SUFFIX_ITEM_TYPE	Suffix item types [sequence of literals: {UNSIGNED_INTEGER, INTEGER, REAL, ...}]
BAND_SUFFIX_BASE	Base values of suffix item scaling [sequence of reals] (see corresponding core element)
BAND_SUFFIX_MULTIPLIER	Multipliers for suffix item scaling [sequence of reals] (see corresponding core element)
BAND_SUFFIX_VALID_MINIMUM	Minimum valid suffix values
BAND_SUFFIX_NULL	...and assigned special values
BAND_SUFFIX_LOW_INSTR_SAT	[sequences of integers or reals]
BAND_SUFFIX_HIGH_INSTR_SAT	(see corresponding core
BAND_SUFFIX_LOW_REPR_SAT	element definitions for
BAND_SUFFIX_HIGH_REPR_SAT	details)

A.23.4 Example

The following label describes ISIS QUBE data from the Galileo NIMS experiment. The QUBE contains 17 bands of NIMS fixed-map mode raw data numbers and 9 backplanes of ancillary information. In other modes, NIMS can produce data qubes of 34, 102, 204 and 408 bands.




```
PDS_VERSION_ID = PDS3
/* File Structure */

RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 512
FILE_RECORDS                   = 9158
LABEL_RECORDS                  =  24
FILE_STATE                     = CLEAN

^HISTORY                       =  25
OBJECT                         = HISTORY
END_OBJECT                     = HISTORY

^QUBE                          = 48
OBJECT                         = QUBE

  /*  Qube structure: Standard ISIS QUBE of NIMS Data */

  AXES                         = 3
  AXIS_NAME                    = (SAMPLE,LINE,BAND)

  /*  Core description */

  CORE_ITEMS                   = (229,291,17)
  CORE_ITEM_BYTES              = 2
  CORE_ITEM_TYPE               = VAX_INTEGER
  CORE_BASE                    = 0.0
  CORE_MULTIPLIER              = 1.0
  CORE_VALID_MINIMUM           = -32752
  CORE_NULL                    = -32768
  CORE_LOW_REPR_SATURATION     = -32767
  CORE_LOW_INSTR_SATURATION    = -32766
```

```
CORE_HIGH_INSTR_SATURATION  = -32765
CORE_HIGH_REPR_SATURATION   = -32764
CORE_NAME                   = RAW_DATA_NUMBER
CORE_UNIT                   = DIMENSIONLESS
PHOTOMETRIC_CORRECTION_TYPE = NONE

/*  Suffix description  */

SUFFIX_BYTES                = 4
SUFFIX_ITEMS                = (0,0,9)
BAND_SUFFIX_NAME            = (LATITUDE,LONGITUDE, INCIDENCE_ANGLE,
    EMISSION_ANGLE, PHASE_ANGLE, SLANT_DISTANCE, INTERCEPT_ALTITUDE,
    PHASE_ANGLE_STD_DEV, RAW_DATA_NUMBER_STD_DEV)
BAND_SUFFIX_UNIT            = (DEGREE, DEGREE, DEGREE, DEGREE,
    DEGREE, KILOMETER, KILOMETER, DEGREE, DIMENSIONLESS)
BAND_SUFFIX_ITEM_BYTES      = (4,4,4,4,4,4,4,4,4)
BAND_SUFFIX_ITEM_TYPE       = (VAX_REAL, VAX_REAL, VAX_REAL,
    VAX_REAL, VAX_REAL, VAX_REAL, VAX_REAL, VAX_REAL, VAX_REAL)
BAND_SUFFIX_BASE            = (0.000000, 0.000000, 0.000000,
    0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000)
BAND_SUFFIX_MULTIPLIER      = (1.000000, 1.000000, 1.000000,
    1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000)
BAND_SUFFIX_VALID_MINIMUM   = (16#FFEFFFFF#, 16#FFEFFFFF#,
    16#FFEFFFFF#, 16#FFEFFFFF#, 16#FFEFFFFF#, 16#FFEFFFFF#,
    16#FFEFFFFF#, 16#FFEFFFFF#, 16#FFEFFFFF#)
BAND_SUFFIX_NULL            = (16#FFFFFFFF#, 16#FFFFFFFF#,
    16#FFFFFFFF#, 16#FFFFFFFF#, 16#FFFFFFFF#, 16#FFFFFFFF#,
    16#FFFFFFFF#, 16#FFFFFFFF#, 16#FFFFFFFF#)
BAND_SUFFIX_LOW_REPR_SAT    = (16#FFFEFFFF#, 16#FFFEFFFF#,
    16#FFFEFFFF#, 16#FFFEFFFF#, 16#FFFEFFFF#, 16#FFFEFFFF#,
    16#FFFEFFFF#, 16#FFFEFFFF#, 16#FFFEFFFF#)
BAND_SUFFIX_LOW_INSTR_SAT   = (16#FFFDFFFF#, 16#FFFDFFFF#,
    16#FFFDFFFF#, 16#FFFDFFFF#, 16#FFFDFFFF#, 16#FFFDFFFF#,
    16#FFFDFFFF#, 16#FFFDFFFF#, 16#FFFDFFFF#)
BAND_SUFFIX_HIGH_INSTR_SAT  = (16#FFFCFFFF#, 16#FFFCFFFF#,
    16#FFFCFFFF#, 16#FFFCFFFF#, 16#FFFCFFFF#, 16#FFFCFFFF#,
    16#FFFCFFFF#, 16#FFFCFFFF#, 16#FFFCFFFF#)
BAND_SUFFIX_HIGH_REPR_SAT   = (16#FFFBFFFF#, 16#FFFBFFFF#,
    16#FFFBFFFF#, 16#FFFBFFFF#, 16#FFFBFFFF#, 16#FFFBFFFF#,
    16#FFFBFFFF#, 16#FFFBFFFF#, 16#FFFBFFFF#)
BAND_SUFFIX_NOTE            = "The backplanes contain 7 geometric
    parameters, the standard deviation   of one of them, the standard
    deviation of a selected data band, and   0 to 10 'spectral index'
    bands, each a user-specified function of the   data bands.  (See
    the BAND_SUFFIX_NAME values.)

    Longitude ranges from 0 to 360 degrees, with positive direction
    specified by POSITIVE_LONGITUDE_DIRECTION in the
    IMAGE_MAP_PROJECTION group.

    INTERCEPT_ALTITUDE contains values for the DIFFERENCE between
    the length of the normal from the center of the target body to
    the line of sight AND the radius of the target body.  On-target
    points   have zero values.  Points beyond the maximum expanded
```

```
    radius have   null values.  This plane thus also serves as a set
    of 'off-limb' flags.  It is meaningful only for the ORTHOGRAPHIC
    and POINT_PERSPECTIVE projections; otherwise all values are zero.
    The geometric standard deviation backplane contains the standard
    deviation of the geometry backplane indicated in its NAME, except
    that the special value 16#FFF9FFFF# replaces the standard
    deviation where the corresponding core pixels have been 'filled'.

    The data band standard deviation plane is computed for the NIMS
    data band specified by STD_DEV_SELECTED_BAND_NUMBER.  This may be
    either a raw data number, or spectral radiance, whichever is
    indicated by CORE_NAME.

    The (optional) spectral index bands were generated by the Vicar
    F2 program. The corresponding BAND_SUFFIX_NAME is an abbreviated
    formula for the function used, where Bn should be read 'NIMS data
    band n'. For example: B4/B8 represents the ratio of bands 4 and
    8."

STD_DEV_SELECTED_BAND_NUMBER   = 9

/*  Data description: general */

DATA_SET_ID                    = "GO-V-NIMS-4-MOSAIC-V1.0"
PRODUCT_ID                     = "XYZ"
SPACECRAFT_NAME                = GALILEO_ORBITER
MISSION_PHASE_NAME             = VENUS_ENCOUNTER
INSTRUMENT_NAME                = NEAR_INFRARED_MAPPING_SPECTROMETER
INSTRUMENT_ID                  = NIMS
^INSTRUMENT_DESCRIPTION        = "NIMSINST.TXT"

TARGET_NAME                    = VENUS
START_TIME                     = 1990-02-10T01:49:58
STOP_TIME                      = 1990-02-10T02:31:52
NATIVE_START_TIME              = 180425.85
NATIVE_STOP_TIME               = 180467.34
OBSERVATION_NAME               = 'VPDIN1'
OBSERVATION_NOTE               = "VPDIN1 / Footprint, Limbfit,
                               Height=50"

INCIDENCE_ANGLE                = 160.48
EMISSION_ANGLE                 = 14.01
PHASE_ANGLE                    = 147.39
SUB_SOLAR_AZIMUTH              = -174.74
SUB_SPACECRAFT_AZIMUTH         = -0.80
MINIMUM_SLANT_DISTANCE         = 85684.10
MAXIMUM_SLANT_DISTANCE         = 103175.00
MIN_SPACECRAFT_SOLAR_DISTANCE = 1.076102e+08
MAX_SPACECRAFT_SOLAR_DISTANCE = 1.076250e+08

/*  Data description: instrument status  */

INSTRUMENT_MODE_ID             = FIXED_MAP
GAIN_MODE_ID                   = 2
```

```
  CHOPPER_MODE_ID               = REFERENCE
  START_GRATING_POSITION        = 16
  OFFSET_GRATING_POSITION       = 04

  MEAN_FOCAL_PLANE_TEMPERATURE  = 85.569702
  MEAN_RAD_SHIELD_TEMPERATURE   = 123.636002
  MEAN_TELESCOPE_TEMPERATURE    = 139.604996
  MEAN_GRATING_TEMPERATURE      = 142.580002
  MEAN_CHOPPER_TEMPERATURE      = 142.449997
  MEAN_ELECTRONICS_TEMPERATURE  = 287.049988

  GROUP                         = BAND_BIN

    BAND_BIN_CENTER             = (0.798777, 0.937873, 1.179840,
      1.458040, 1.736630, 2.017250, 2.298800, 2.579060, 2.864540,
      3.144230, 3.427810, 3.710640, 3.993880, 4.277290, 4.561400,
      4.843560, 5.126080)
    BAND_BIN_UNIT               = MICROMETER
    BAND_BIN_ORIGINAL_BAND      = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                  12, 13, 14, 15, 16, 17)
    BAND_BIN_GRATING_POSITION   = (16, 16, 16, 16, 16, 16, 16, 16, 16,
                                  16, 16, 16, 16, 16, 16, 16, 16)
    BAND_BIN_DETECTOR           = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                  12, 13, 14, 15, 16, 17)
  END_GROUP                     = BAND_BIN

  GROUP                         = IMAGE_MAP_PROJECTION
    /* Projection description */
    MAP_PROJECTION_TYPE         = OBLIQUE_ORTHOGRAPHIC
    MAP_SCALE                   = 45.000
    MAP_RESOLUTION              = 2.366
    CENTER_LATITUDE             = 12.00
    CENTER_LONGITUDE            = 350.00
    LINE_PROJECTION_OFFSET      = 149.10
    SAMPLE_PROJECTION_OFFSET    = 85.10
    MINIMUM_LATITUDE            = 11.71
    MAXIMUM_LATITUDE            = 13.62
    MINIMUM_LONGITUDE           = 349.62
    MAXIMUM_LONGITUDE           = 351.72
    POSITIVE_LONGITUDE_DIRECTION= EAST
    A_AXIS_RADIUS               = 6101.000000
    B_AXIS_RADIUS               = 6101.000000
    C_AXIS_RADIUS               = 6101.000000
    REFERENCE_LATITUDE          = 0.000000
    REFERENCE_LONGITUDE         = 0.000000
    MAP_PROJECTION_ROTATION     = 0.00
    LINE_FIRST_PIXEL            = 1
    LINE_LAST_PIXEL             = 229
    SAMPLE_FIRST_PIXEL          = 1
    SAMPLE_LAST_PIXEL           = 291
  END_GROUP                     = IMAGE_MAP_PROJECTION

END_OBJECT                      = QUBE
END
```

A.24 SERIES

The SERIES object is a sub-class of the TABLE object. It is used for storing a sequence of measurements organized in a specific way (e.g., chronologically, by radial distance, etc.). The SERIES uses the same physical format specification as the TABLE object with additional sampling parameter information describing the variation between elements in the series. The sampling parameter keywords are required for the SERIES object itself, but are optional for the COLUMN sub-objects, depending on the data organization.

The sampling parameter keywords in the SERIES object represent the variation between the ROWS of data. For data with regularly-spaced rows, the SAMPLING_PARAMETER_INTERVAL keyword defines the row-to-row variation. For data in which rows are irregularly spaced, the SAMPLING_PARAMETER_INTERVAL keyword is "N/A" and the actual sampling parameter is included as a COLUMN in the SERIES.

When the data vary regularly across items of a single column, sampling parameter keywords appear as part of the COLUMN sub-object. Data sampled at irregular intervals described as separate columns may also provide sampling parameter information specific to each column.

Optional MINIMUM_SAMPLING_PARAMETER and MAXIMUM_SAMPLING_PARAMETER keywords should be added whenever possible to indicate the range in which the data were sampled. For data sampled at a single point rather than over a range, both the MINIMUM_SAMPLING_PARAMETER and MAXIMUM_SAMPLING_PARAMETER are set to the specific value.

The object name "TIME_SERIES" is used when the series is chronological. In this case the label keywords START_TIME and STOP_TIME are assumed to indicate the minimum and maximum times in the file. If this is not the case, the MINIMUM_SAMPLING_PARAMETER and MAXIMUM_SAMPLING_PARAMETER keywords should be used to specify the corresponding time values for the series.

A.24.1 Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES
5. SAMPLING_PARAMETER_NAME
6. SAMPLING_PARAMETER_UNIT
7. SAMPLING_PARAMETER_INTERVAL

A.24.2 Optional Keywords

1. NAME
2. ROW_PREFIX_BYTES
3. ROW_SUFFIX_BYTES
4. MINIMUM_SAMPLING_PARAMETER
5. MAXIMUM_SAMPLING_PARAMETER
6. DERIVED_MINIMUM
7. DERIVED_MAXIMUM
8. DESCRIPTION

A.24.3 Required Objects

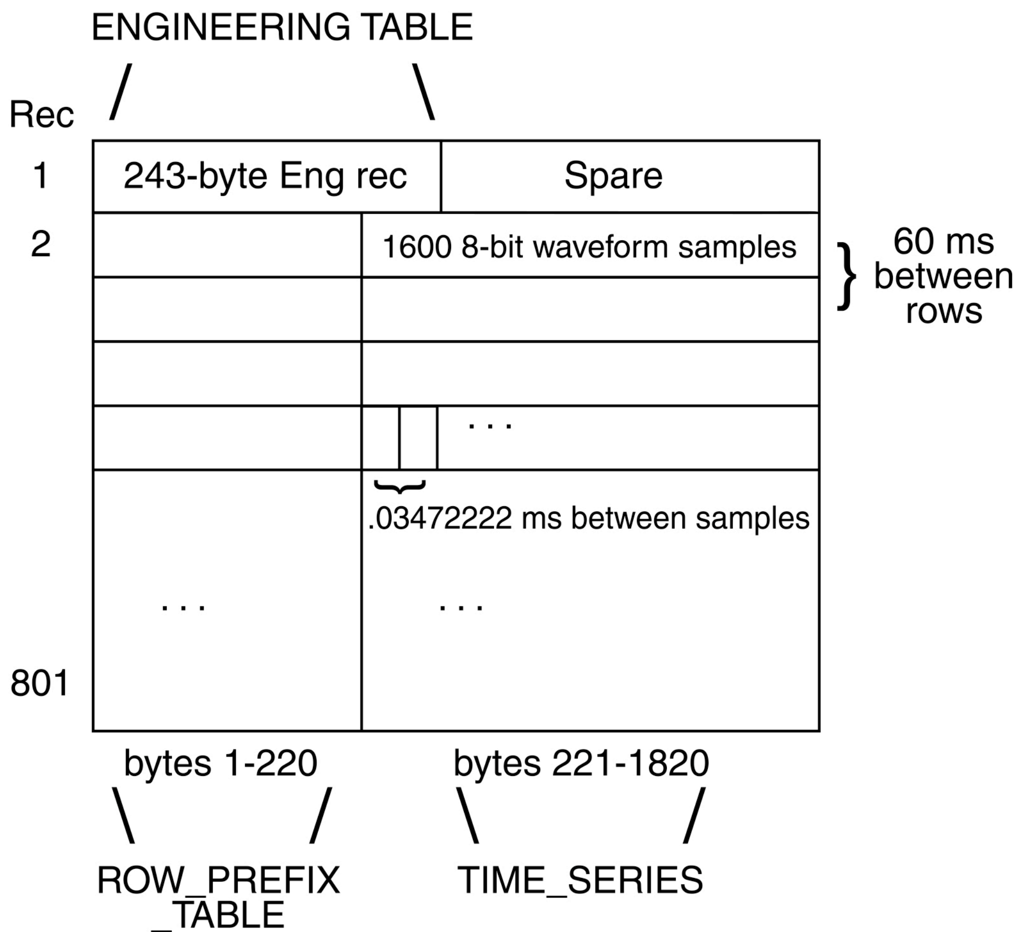
1. COLUMN

A.24.4 Optional Objects

1. CONTAINER

A.24.5 Example

This example illustrates the use of the SERIES object for data that vary regularly in two ways: rows of data in the SERIES occur at 60 millisecond intervals, while the column values occur at .03472222 millisecond intervals. Note that, as with other forms of the TABLE object, each row in a SERIES may contain prefix or suffix bytes, indicated in this case by the ROW_PREFIX_BYTES in the TIME_SERIES definition. The structure of the prefix is defined by the ROW_PREFIX_TABLE object, for which the COLUMN definitions are stored in a separate file (“ROWPRX.FMT”).




```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 1820
FILE_RECORDS                    = 801
^ENGINEERING_TABLE              = ("C0900313.DAT", 1)
^ROW_PREFIX_TABLE               = ("C0900313.DAT", 2)
^TIME_SERIES                    = ("C0900313.DAT", 2)

/* Observation description */
DATA_SET_ID                     = "VG2-N-PWS-2-EDR-WFRM-60MS-V1.0"
PRODUCT_ID                      = "C0900313.DAT"
PRODUCT_CREATION_TIME           = "UNK"
SPACECRAFT_NAME                 = VOYAGER_2
SPACECRAFT_CLOCK_START_COUNT    = "09003.13.002"
SPACECRAFT_CLOCK_STOP_COUNT     = "09003.13.002"
EARTH_RECEIVED_TIME             = 1989-159T13:35:00.121
START_TIME                      = 1989-157T14:16:56.979
STOP_TIME                       = "N/A"
MISSION_PHASE_NAME              = NEPTUNE_ENCOUNTER
TARGET_NAME                     = NEPTUNE
```

```
/* Instrument description */
INSTRUMENT_NAME                = PLASMA_WAVE_RECEIVER
INSTRUMENT_ID                  = PWS
SECTION_ID                     = WFRM

/* Object descriptions */
OBJECT                         = ENGINEERING_TABLE
  INTERCHANGE_FORMAT           = BINARY
  ROWS                         = 1
  COLUMNS                      = 106
  ROW_BYTES                    = 243
  ROW_SUFFIX_BYTES             = 1577
  DESCRIPTION                  = "This table describes the format of
    the engineering record which is included as the first record in
    each PWS high rate waveform file.  This record contains the  first
    242 bytes of data extracted from the Mission and Test Imaging System
    (MTIS) header record on each file of an imaging EDR tape.  A 243rd
    byte containing some flag fields has been added to the table for all
    data collected during the Neptune encounter."
  ^STRUCTURE                   = "ENGTAB.FMT"
END_OBJECT                     = ENGINEERING_TABLE

OBJECT                         = ROW_PREFIX_TABLE
  INTERCHANGE_FORMAT           = BINARY
  ROWS                         = 800
  COLUMNS                      = 47
  ROW_BYTES                    = 220
  ROW_SUFFIX_BYTES             = 1600
  DESCRIPTION                  = "This table describes the format of
    the engineering data associated with the collection of each row of
    waveform data (1600 waveform samples)."
  ^STRUCTURE                   = "ROWPRX.FMT"
END_OBJECT                     = ROW_PREFIX_TABLE

OBJECT                         = TIME_SERIES
  NAME                         = WAVEFORM_FRAME
  INTERCHANGE_FORMAT           = BINARY
  ROWS                         = 799
  COLUMNS                      = 1
  ROW_BYTES                    = 1600
  ROW_PREFIX_BYTES             = 220
  SAMPLING_PARAMETER_NAME      = TIME
  SAMPLING_PARAMETER_UNIT      = SECOND
  SAMPLING_PARAMETER_INTERVAL  = .06         /* 60 MS between rows */
  DESCRIPTION                  = "This time_series consists of up to
    800  records (or rows, lines) of PWS waveform sample data.  Each
    record 2-801 of the file (or frame) contains 1600 waveform samples,
    prefaced by 220 bytes of MTIS information.  The 1600 samples are
    collected in 55.56 msec followed by a 4.44 msec gap.  Each 60 msec
    interval constitutes a line of waveform samples.  Each file contains
    up to 800 lines of waveform samples for a 48 sec frame."

  OBJECT                       = COLUMN
    NAME                       = WAVEFORM_SAMPLES
```

```
    DATA_TYPE                    = MSB_UNSIGNED_INTEGER
    START_BYTE                   = 221
    BYTES                        = 1600
    ITEMS                        = 1600
    ITEM_BYTES                   = 1
    SAMPLING_PARAMETER_NAME      = TIME
    SAMPLING_PARAMETER_UNIT      = SECOND
    SAMPLING_PARAMETER_INTERVAL  = 0.00003472222 /*time between
                                   samples*/

    OFFSET                       = -7.5
    VALID_MINIMUM                = 0
    VALID_MAXIMUM                = 15
    DESCRIPTION                  = "The 1-byte waveform samples
    constitute an array of waveform measurements which are encoded into
    binary values from 0 to 15 and may be re-mapped to reduce the
    artificial zero-frequency component. For example,  stored values can
    be mapped to the following floating point  values.  The original 4-
    bit data samples have been repackaged into 8-bit (1 byte) items
    without modification for archival purposes.\n

                     0  = -7.5  1  = -6.5 2  = -5.5     3  = -4.5
                     4  = -3.5  5  = -2.5 6  = -1.5     7  = -0.5
                     8  =  0.5  9  =  1.5 10 =  2.5     11 =  3.5
                     12 =  4.5  13 =  5.5 14 =  6.5     15 =  7.5
       "
  END_OBJECT                     = COLUMN
END_OBJECT                       = TIME_SERIES

END
```

A.25 SPECTRAL_QUBE

A.25.1 Introduction

Instruments classified as imaging spectrometers are increasingly being used in planetary missions. Data from these instruments are simultaneously a set of images, at different wavelengths, of the same target area, and a set of spectra at each point of the target area. In PDS archives, these data may be stored as SPECTRAL_QUBEs, three-dimensional objects with two spatial dimensions and one spectral dimension. In these three-dimensional structures, called "qubes", the axes have the interpretations "sample", "line", and "band", respectively.

Each of the three axes in a PDS SPECTRAL_QUBE object may optionally include suffix data that extend the length of the axis. Conceptually, this can be viewed as forming one or more Suffix planes that are attached to the Core qube, as shown in the diagram below. Suffix planes that extend the band dimension are called BACKPLANES. Suffix planes that extend the sample dimension are called SIDEPLANES. Suffix planes that extend the line dimension are called BOTTOMPLANES.

Note that these terms refer to the "logical" axes – that is, how the axes are conceptually modeled – and are not necessarily related to the physical storage of the SPECTRAL_QUBE object. The Suffix planes are used for storing auxiliary data that are associated with the core data. For example, a backplane might be used for storing the latitude values for each spatial-spatial pixel. Another backplane might be used for storing the wavelength of the deepest absorption feature that was found in the spectrum at each spatial-spatial pixel. One or more SIDEPLANES might be used for storing engineering data that are associated with each spatial line.

A.25.2 Logical Structure of a SPECTRAL_QUBE

As mentioned above, the logical structure of the SPECTRAL_QUBE is its conceptual model. This is best presented visually, as is shown in the following diagrams;

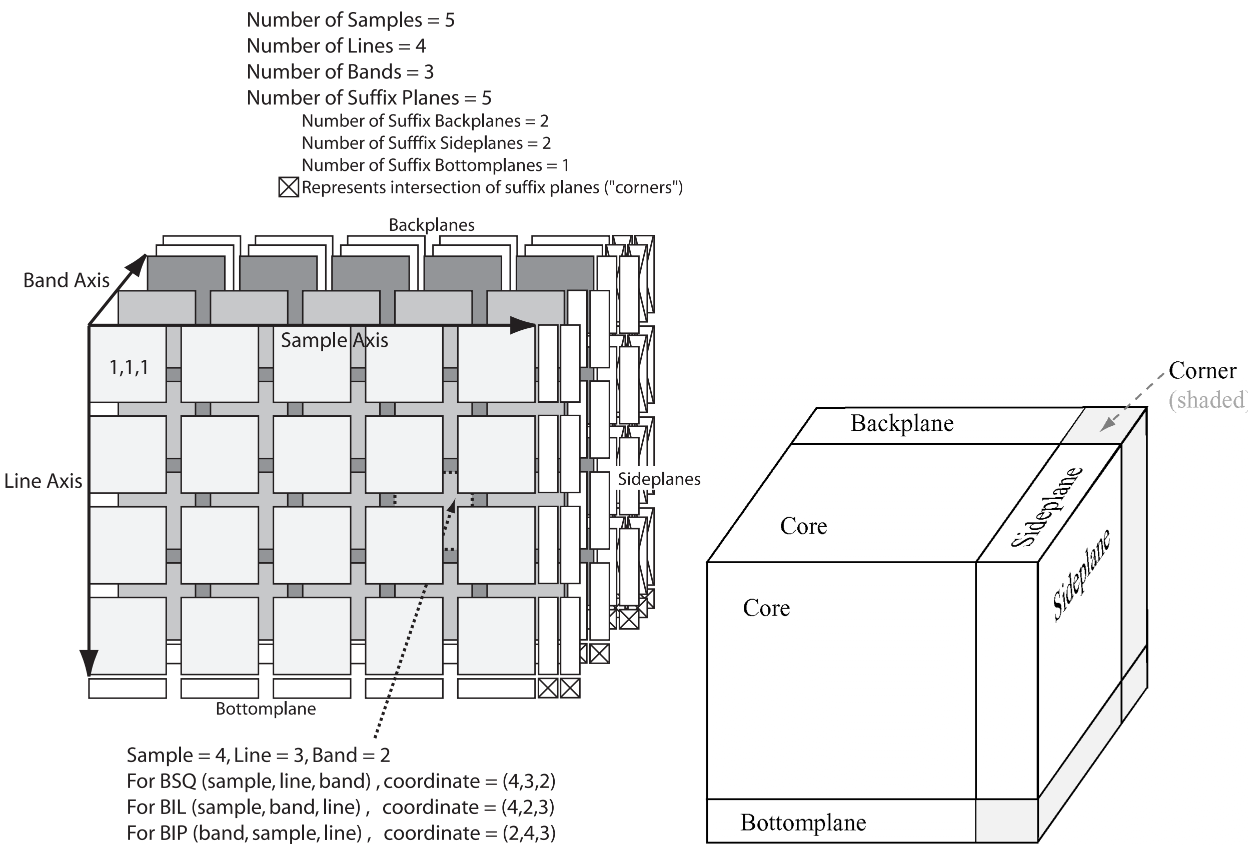


Figure A.4 – Exploded Views of a SPECTRAL_QUBE Object

A.25.2.1 Pixel Coordinates

SAMPLE=1 is the left edge of the spatial-spatial core image. LINE=1 is the top edge of the spatial-spatial core image. BAND=1 corresponds to the spatial-spatial images at the “front” of the diagram. Core coordinates do not carry over to the suffix regions.

A.25.3 Physical Structure of a SPECTRAL_QUBE

A.25.3.1 Storage Orders

The file in which a PDS SPECTRAL_QUBE data object is stored is physically accessed as though it were a one-dimensional data structure. Storing the PDS SPECTRAL_QUBE pictured above thus requires that the “logical” three-dimensional structure be mapped into the one-dimensional physical file structure. This involves moving through the three-dimensional structure in certain patterns to determine the linear sequence of core and suffix pixel values that occur in the file. In PDS SPECTRAL_QUBE files, this pattern is defined by specifying which axis index varies fastest in the linear sequence of pixel values in the file, which axis varies second fastest, and which axis varies slowest.

In PDS SPECTRAL_QUBE files, the names of the three axes are always SAMPLE, LINE, and BAND. The AXIS_NAME keyword has an array of values that list the names of the axes in the qube. The order of the names specifies the qube storage order in the file. The first axis is the fastest varying, and the third axis is the slowest varying. The PDS SPECTRAL_QUBE supports the following three storage orders:

- (SAMPLE, LINE, BAND) – Band Sequential (BSQ)
- (SAMPLE, BAND, LINE) – Band Interleaved by Line (BIL)
- (BAND, SAMPLE, LINE) – Band Interleaved by Pixel (BIP)

The lengths of the Core axes are given by the CORE_ITEMS keyword, and the lengths of the Suffix axes are given by the SUFFIX_ITEMS keyword. Both these keywords have array values, whose order corresponds to the order of the axes given by the AXIS_NAME keyword.

In the physical file storage, Suffix pixel data (if present) are interspersed with the associated Core pixel data. For example, in a BSQ storage order file, the physical qube storage in the file begins with the pixels in the first (top) line of the spatial-spatial image plane at the first wavelength band. This is followed by the sideplane pixel values that extend this line of core pixels. Next are the core pixels for the second line, followed by the sideplane pixels for the second line. After the last line of this first core image plane (and its associated sideplane pixels) come the bottomplane pixels associated with the first band. This is then repeated for the second through last bands. Finally, all the backplane data are stored after all the core data and associated sideplane and bottomplane pixels.

If a PDS SPECTRAL_QUBE file includes suffixes on more than one axis, then the region that is the intersection between two (or all three) of the suffix regions is called a CORNER region. The PDS requires that space for CORNER region data be allocated in the data files. However this space is never actually used.

A.25.3.2 Pixel Storage Sizes

In a PDS SPECTRAL_QUBE file, core pixels can occupy one, two, or four bytes. All core pixels within a single file must be of the same physical storage size. Suffix pixels can also occupy one, two, or four bytes of storage in the file. All the suffix pixels within a single file must be of the same physical storage size. Suffix pixels need not be the same size as core pixels. *Handling of different pixel data types is described in detail below.*

A.25.3.3 Core Pixel Data Types

In PDS SPECTRAL_QUBE files, core pixel values can be represented by one of several formats. The formats available are dependent on the number of bytes used to store the values in the file. The format is given by the CORE_ITEM_TYPE keyword and the number of bytes stored is given by the CORE_ITEM_BYTES keyword. The following table shows the allowable formats and the number of bytes of storage they use:

CORE_ITEM_BYTES	CORE_ITEM_TYPE	Type Conversion Parameters
1, 2, or 4	UNSIGNED_INTEGER	Yes
1, 2, or 4	MSB_UNSIGNED_INTEGER	Yes
1, 2, or 4	LSB_UNSIGNED_INTEGER	Yes
1, 2, or 4	INTEGER	Yes
1, 2, or 4	MSB_INTEGER	Yes
1, 2, or 4	LSB_INTEGER	Yes
4	IEEE_REAL	No
4	VAX_REAL	No
4	PC_REAL	No

As the table above indicates, stored integer values can be converted to real values, representing the actual pixel. The type conversion parameters are given by the CORE_BASE and CORE_MULTIPLIER keywords, and the real value being represented is determined as follows:

“real_value” = CORE_BASE + (CORE_MULTIPLIER * REAL(stored_value))

For 4-byte real formats, the stored values are floating point values that directly represent the pixel values.

A.25.3.4 Suffix Pixel Data Types

The same data types and number of storage bytes that are shown in the above table are also available to Suffix pixels. However, Suffix pixels need not be the same size or have the same data type as the Core pixels. Therefore, there is a SUFFIX_ITEM_BYTES keyword to indicate the number of bytes stored for Suffix pixels and a SUFFIX_ITEM_TYPE keyword to describe the data type of the Suffix pixels. Each suffix plane within a single file can have a different data format. Thus, the values of these keywords are arrays. Each element of the array refers to a spearate suffix plane.

A.25.3.5 Aligning Suffix Pixels within Allocated Bytes

The SPECTRAL_QUBE allows the number of bytes used to store data in each Suffix pixel (SUFFIX_ITEM_BYTES) to be less than the total number of bytes allocated to each Suffix pixel

(SUFFIX_BYTES). It is therefore necessary to describe how the stored bytes are aligned within the allocated bytes. The BIT_MASK keyword is used for this purpose.

A.25.4 Data Dictionary Elements for the SPECTRAL_QUBE

The following section details the required and optional data dictionary elements that comprise the SPECTRAL_QUBE.

NOTE: Some of the following required and optional elements of the SPECTRAL_QUBE object are ISIS-specific. Since the ISIS system was designed before the current version of the Planetary Science Data Dictionary, some of the element names below conflict with current PDS nomenclature standards.

A.25.4.1 Required Objects

None.

A.25.4.2 Optional Objects

Object Name	Definition
IMAGE_MAP_PROJECTION	Map projection information for the image planes.

A.25.4.3 Required Groups

Group Name	Definition
BAND_BIN	Group describing properties of each “bin” along the spectral axis.

A.25.4.4 Optional Groups

The following groups are optional, in that they describe optional Suffix axes. However, if the named axis does appear, its descriptive keywords must be part of the appropriate group:

Group Name	Definition
BAND_SUFFIX	Group describing properties of the BAND Suffix plane (“BACKPLANE”).

LINE_SUFFIX	Group describing properties of the LINE Suffix plane ("BOTTOMPLANE").
SAMPLE_SUFFIX	Group describing properties of the SAMPLE Suffix plane ("SIDEPLANE").

A.25.4.5 Required Keywords – Outside of Groups

Keyword Name	Definition	Values
AXES	Number of axes or dimensions of SPECTRAL_QUBE	3 (SPECTRAL_QUBEs are 3-dimensional by definition).
AXIS_NAME	Names of axes in order of physical storage.	Literal values SAMPLE, LINE, and BAND in storage order. One of these three storage orders is required: (SAMPLE, LINE, BAND) (BAND, SAMPLE, LINE) (SAMPLE, BAND, LINE).
CORE_ITEMS	Number of pixels on each axis of the Core, in the same order as in AXIS_NAME	Sequence of three integers, e.g. (256, 512, 3).
CORE_ITEM_BYTES	Number of bytes in each core pixel.	1, 2, or 4.
CORE_ITEM_TYPE	Data type of core pixels.	UNSIGNED_INTEGER, MSB_UNSIGNED_INTEGER, LSB_UNSIGNED_INTEGER, INTEGER, MSB_INTEGER, LSB_INTEGER, IEEE_REAL, VAX_REAL, PC_REAL.
SUFFIX_ITEMS	Number of side (SAMPLE) suffix planes, bottom (LINE) suffix planes, and back (BAND) suffix planes, in same order as in AXIS_NAME.	Sequence of three integers. If there are no suffix planes, the value is (0, 0, 0).
If suffix planes are present:		
SUFFIX_BYTES	Number of bytes allocated for each suffix pixel.	1, 2, or 4. See also SUFFIX_ITEM_BYTES.

A.25.4.6 Required Keywords – In the *_SUFFIX Groups

If there are SUFFIX planes, then the following keywords are required. In order to avoid having to create up to three instances of each one (e.g., BAND_SUFFIX_NAME, LINE_SUFFIX_NAME, and SAMPLE_SUFFIX_NAME), the keywords must be nested in the appropriate group (see section on Optional Groups):

BAND_SUFFIX group – if describing a BAND SUFFIX
LINE_SUFFIX group – if describing a LINE SUFFIX
SAMPLE_SUFFIX group – if describing a SAMPLE SUFFIX

Keyword Name	Definition	Values
SUFFIX_NAME	Name of suffix plane	Literal, e.g. LATITUDE
SUFFIX_ITEM_BYTES	Number of bytes used to store data in each suffix pixel; may be less than the number of bytes allocated for each pixel.	1, 2, or 4. See also SUFFIX_BYTES.
SUFFIX_ITEM_TYPE	Data type of suffix pixels.	UNSIGNED_INTEGER, MSB_UNSIGNED_INTEGER, LSB_UNSIGNED_INTEGER, INTEGER, MSB_INTEGER, LSB_INTEGER, IEEE_REAL, VAX_REAL, PC_REAL.

A.25.4.7 Required Keywords – In the BAND_BIN Group

Keyword Name	Definition	Values
BANDS	Number of bands in SPECTRAL_QUBE (same as given for the BAND axis in CORE_ITEMS, repeated here for convenience).	Integer.
BAND_BIN_CENTER	Wavelengths or frequencies at band centers.	Sequence of real values, one per band.
BAND_BIN_UNIT	Unit of measurement of BAND_BIN_CENTER and BAND_BIN_WIDTH values.	For example, MICROMETER.
BAND_BIN_WIDTH	Widths (at half height) of bands.	Sequence of real values, one per band.

Note: In the case where there are so many bands that the BAND_BIN group becomes cumbersome in the label, it may be stored in a separate file indicated in the label by a structure pointer, e.g. ^STRUCTURE = “BAND_BIN.FMT”.

A.25.4.8 Optional Keywords

The following keywords are optional for the PDS SPECTRAL_QUBE. Some of these keywords must be used if the SPECTRAL_QUBE is designed for use with the Integrated Software for Imagers and Spectrometers (ISIS). The column labeled ISIS indicates whether the keyword is required by ISIS software. A “YES” means the keyword is required by ISIS, while a “NO” means it is not:

Keyword Name	Definition	Values	ISIS
ISIS_STRUCTURE_VERSION	Version of ISIS software with which the SPECTRAL_QUBE’s physical structure is compatible.	2.1 (Only current valid version number)	YES
CORE_NAME	Name of data value stored in the SPECTRAL_QUBE	Literal, e.g. SPECTRAL_RADIANCE.	YES
CORE_BASE	Base value for scaling core pixels.	Real.	YES
CORE_MULTIPLIER	Multiplier for scaling core pixels.	Real.	YES
CORE_UNIT	Unit of measurement of core data values.	For example, ‘“WATT*M**-2*SR**-1*mM**-1’ (for spectral radiance) or ‘DIMENSIONLESS’ (for raw data).	YES
CORE_VALID_MINIMUM	Minimum valid core value.	Values below CORE_VALID_MINIMUM have special meaning.	YES
CORE_NULL	Special value that indicates invalid data.	Must be less than CORE_VALID_MINIMUM.	YES
CORE_LOW_REPR_ SATURATION	Special value that indicates representation saturation at low end.	Must be less than CORE_VALID_MINIMUM.	YES
CORE_LOW_INSTR_ SATURATION	Special value that indicates instrument saturation at low end.	Must be less than CORE_VALID_MINIMUM.	YES
CORE_HIGH_REPR_ SATURATION	Special value that indicates representation saturation at high end.	Must be less than CORE_VALID_MINIMUM.	YES

CORE_HIGH_INSTR_ SATURATION	Special value that indicates instrument saturation at high end.	Must be less than CORE_VALID_MINIMUM.	YES
SUFFIX_BYTES	Number of bytes allocated for each suffix pixel (required even if no suffix planes are present).	1, 2, or 4. See also SUFFIX_ITEM_BYTES.	YES
MD5_CHECKSUM	MD5 checksum of all core and suffix bytes.	Character String.	NO
LINE_DISPLAY_DIRECTION	The preferred orientation of lines within an image for viewing on a display device. The default value is down, where lines are viewed top to bottom on the display.	DOWN, UP, LEFT, RIGHT. (Value must be orthogonal to value for SAMPLE_-DISPLAY_DIRECTION.)	NO
SAMPLE_DISPLAY_DIRECTION	The preferred orientation of samples within a line for viewing on a display device. The default is right, meaning samples are viewed from left to right on the display.	DOWN, UP, LEFT, RIGHT. (Value must be orthogonal to value for LINE_DISPLAY_-DIRECTION.)	NO
In BAND_SUFFIX, LINE_SUFFIX, and SAMPLE_SUFFIX groups:			
BIT_MASK	A series of binary digits defining the active bits in a value. Required when fewer bytes are used than are allocated.	A sequence of bits equal to the bit-length of the allocated storage.	NO
SUFFIX_BASE	Base value for scaling suffix pixels.	Real.	NO
SUFFIX_MULTIPLIER	Multiplier for scaling suffix pixels.	Real.	NO
SUFFIX_VALID_MINIMUM	Minimum valid suffix value.	Values below SUFFIX_VALID_MINIMUM have special meaning.	NO
SUFFIX_NULL	Special value that indicates invalid data.	Must be less than SUFFIX_VALID_MINIMUM.	NO
SUFFIX_LOW_REPR_SAT	Special value that indicates representation saturation at low end.	Must be less than SUFFIX_VALID_MINIMUM.	NO
SUFFIX_LOW_INSTR_SAT	Special value that indicates instrument saturation at low end.	Must be less than SUFFIX_VALID_MINIMUM.	NO
SUFFIX_HIGH_REPR_SAT	Special value that indicates representation saturation at high end.	Must be less than SUFFIX_VALID_MINIMUM.	NO

SUFFIX_HIGH_INSTR_SAT	Special value that indicates instrument saturation at high end.	Must be less than SUFFIX_VALID_MINIMUM.	NO
SUFFIX_UNIT	Unit of measurement of suffix data values.	For example, 'DEGREE', 'DIMENSIONLESS'.	NO
In BAND_BIN group:			
BAND_BIN_STANDARD_ DEVIATION	Standard deviations of spectrometer values at each band.	Sequence of real values, one per band.	NO
BAND_BIN_DETECTOR	Instrument detector number of each band, where relevant.	Sequence of integers, one per band.	NO
BAND_BIN_GRATING_ POSITION	Instrument grating position of each band, where relevant.	Sequence of integers, one per band.	NO
BAND_BIN_ORIGINAL_BAND	Where relevant, band numbers from the original qube of which the current qube is a subset. Band numbers in the original qube are sequential integers.	Sequence of integers, one per band, listed in storage order for the current qube.	NO
BAND_BIN_BAND_NUMBER	List of band numbers corresponding to each band contained in the image. The band number is equivalent to the instrument band number.	Sequence of integers, one per band.	NO
BAND_BIN_FILTER_NUMBER	List of filter numbers corresponding to each band contained in the image. The filter number describes the physical location of the band in the detector array. Filter 1 is on the leading edge of the array.	Sequence of integers, one per band.	NO
BAND_BIN_BASE	The offset value for the stored data of each band listed in the BAND_BIN_BAND_NUMBER. The BAND_BIN_BASE value is added to the scaled data (see BAND_BIN_MULTIPLIER) to reproduce the true data.	Sequence of real values, one per band.	NO
BAND_BIN_MULTIPLIER	The constant value by which the stored data of each band listed in the BAND_BIN_BAND_NUMBER is multiplied to produce the scaled data; the BAND_BIN_BASE value is added to the scaled data to reproduce the true data.	Sequence of real values, one per band.	NO

Example label for a PDS SPECTRAL_QUBE

```
PDS_VERSION_ID                    = PDS3

/* File Identification and Structure */

RECORD_TYPE                       = FIXED_LENGTH
RECORD_BYTES                      = 644
FILE_RECORDS                      = 249888

/* Pointer to Data Object */

^SPECTRAL_QUBE                    = "SAMPLE1.QUB"

/* Identification Data Elements */

DATA_SET_ID                     =
PRODUCT_ID                     =
INSTRUMENT_HOST_NAME              =
INSTRUMENT_NAME                 =
TARGET_NAME                     =
START_TIME                      =
STOP_TIME                      =
SPACECRAFT_CLOCK_START_COUNT      =
SPACECRAFT_CLOCK_STOP_COUNT        =
PRODUCT_CREATION_TIME            =

/* SPECTRAL_QUBE Object Description */

OBJECT                         = SPECTRAL_QUBE

  AXES                            = 3
  AXIS_NAME                       = (SAMPLE, LINE, BAND)
  ISIS_STRUCTURE_VERSION          = "N/A"
  MD5_CHECKSUM                    = cf65a98aff4232f5ac5171406590a932

/* Core Description */

  CORE_ITEMS                      = (320, 272, 224)
  CORE_NAME                       = "CALIBRATED SPECTRAL RADIANCE"
  CORE_ITEM_BYTES                 = 2
  CORE_ITEM_TYPE                  = MSB_INTEGER
  CORE_BASE                       = 0.000000
  CORE_MULTIPLIER                 = 1.000000
  CORE_UNIT                       = "WATT*CM**-2*SR**-1*UM**-1"
  CORE_NULL                       = -32768
  CORE_VALID_MINIMUM              = -32752
  CORE_LOW_REPR_SATURATION        = -32767
  CORE_LOW_INSTR_SATURATION       = -32766
  CORE_HIGH_REPR_SATURATION       = -32765
  CORE_HIGH_INSTR_SATURATION      = -32764
```

```
/* Suffix Descriptions */

  SUFFIX_ITEMS                  = (1, 1, 2)
  SUFFIX_BYTES                  = 4

  GROUP                         = SAMPLE_SUFFIX
    SUFFIX_NAME                 = HORIZONTAL_DESTRIPE
    SUFFIX_ITEM_BYTES           = 4
    SUFFIX_ITEM_TYPE            = IEEE_REAL
    SUFFIX_BASE                 = 0.000000
    SUFFIX_MULTIPLIER           = 1.000000
    SUFFIX_VALID_MINIMUM        = 16#FFEFFFFF#
    SUFFIX_NULL                 = 16#FFFFFFFF#
    SUFFIX_LOW_REPR_SAT         = 16#FFFEFFFF#
    SUFFIX_LOW_INSTR_SAT        = 16#FFFDFFFF#
    SUFFIX_HIGH_REPR_SAT        = 16#FFFBFFFF#
    SUFFIX_HIGH_INSTR_SAT       = 16#FFFCFFFF#
  END_GROUP                     = SAMPLE_SUFFIX

  GROUP                         = LINE_SUFFIX
    SUFFIX_NAME                 = VERTICAL_DESTRIPE
    SUFFIX_ITEM_BYTES           = 4
    SUFFIX_ITEM_TYPE            = IEEE_REAL
    SUFFIX_BASE                 = 0.000000
    SUFFIX_MULTIPLIER           = 1.000000
    SUFFIX_VALID_MINIMUM        = 16#FFEFFFFF#
    SUFFIX_NULL                 = 16#FFFFFFFF#
    SUFFIX_LOW_REPR_SAT         = 16#FFFEFFFF#
    SUFFIX_LOW_INSTR_SAT        = 16#FFFDFFFF#
    SUFFIX_HIGH_REPR_SAT        = 16#FFFBFFFF#
    SUFFIX_HIGH_INSTR_SAT       = 16#FFFCFFFF#
  END_GROUP                     = LINE_SUFFIX

  GROUP                         = BAND_SUFFIX
    SUFFIX_NAME                 = (LATITUDE, LONGITUDE)
    SUFFIX_UNIT                 = (DEGREE, DEGREE)
    SUFFIX_ITEM_BYTES           = (4, 4)
    SUFFIX_ITEM_TYPE            = (IEEE_REAL, IEEE_REAL)
    SUFFIX_BASE                 = (0.000000, 0.000000)
    SUFFIX_MULTIPLIER           = (1.000000, 1.000000)
  END_GROUP                     = BAND_SUFFIX

  /* Band bin information */
  /* For this example with 224 bands: */
  /* The BAND_BIN group is stored in a separate file. */

  ^STRUCTURE                    = “BAND_BIN.FMT”

  /* Map projection information */

  OBJECT                        = IMAGE_MAP_PROJECTION
    A_AXIS_RADIUS               = 1737.4000000
    B_AXIS_RADIUS               = 1737.4000000
    C_AXIS_RADIUS               = 1737.4000000
```

```
    POSITIVE_LONGITUDE_DIRECTION = EAST
    MAP_PROJECTION_TYPE          = “SINUSOIDAL EQUAL AREA”
    MAP_SCALE                    = 0.1000000
    MAP_RESOLUTION               = 303.2334900
    EASTERNMOST_LONGITUDE        = 126.0177002
    WESTERNMOST_LONGITUDE        = 120.0000000
    MINIMUM_LATITUDE             = 20.9867992
    MAXIMUM_LATITUDE             = 28.0000000
    CENTER_LONGITUDE             = 135.0000000
    REFERENCE_LATITUDE           = 0.0000000
    REFERENCE_LONGITUDE          = 0.0000000
    MAP_PROJECTION_ROTATION      = 0.0000000
    LINE_PROJECTION_OFFSET       = -8490.0381188
    SAMPLE_PROJECTION_OFFSET     = -4246.2684059
  END_OBJECT                     = IMAGE_MAP_PROJECTION

END_OBJECT                       = SPECTRAL_QUBE

END
```

A.25.5 Contents of Example BAND_BIN.FMT

```
GROUP = BAND_BIN
  BANDS = 224
  BAND_BIN_UNIT = MICROMETER
  BAND_BIN_CENTER = (
0.374370, 0.384460, 0.394120, 0.403770, 0.413430, 0.423090, 0.432750,
0.442420, 0.452080, 0.461750, 0.471410, 0.481080, 0.490750, 0.500410,
0.510080, 0.519760, 0.529430, 0.539100, 0.548780, 0.558450, 0.568130,
0.577810, 0.587490, 0.597170, 0.606850, 0.616530, 0.626210, 0.635900,
0.645580, 0.655270, 0.664960, 0.676310, 0.655020, 0.664890, 0.674430,
0.683970, 0.693520, 0.703070, 0.712620, 0.722170, 0.731730, 0.741290,
0.750860, 0.760420, 0.770000, 0.779570, 0.789150, 0.798720, 0.808310,
0.817890, 0.827480, 0.837070, 0.846670, 0.856270, 0.865870, 0.875470,
0.885080, 0.894690, 0.904300, 0.913920, 0.923540, 0.931740, 0.946990,
0.956410, 0.966100, 0.975560, 0.985010, 0.994470, 1.003930, 1.013390,
1.022840, 1.032300, 1.041760, 1.051210, 1.060670, 1.070130, 1.079590,
1.089040, 1.098500, 1.107950, 1.117410, 1.126870, 1.136320, 1.145780,
1.155240, 1.164690, 1.174150, 1.183600, 1.193060, 1.202520, 1.211970,
1.221430, 1.230890, 1.240340, 1.249800, 1.259770, 1.254350, 1.264320,
1.274300, 1.284270, 1.294240, 1.304210, 1.314180, 1.324150, 1.334130,
1.344100, 1.354070, 1.364040, 1.374010, 1.383980, 1.393950, 1.403920,
1.413880, 1.423850, 1.433820, 1.443790, 1.453760, 1.463720, 1.473690,
1.483660, 1.493620, 1.503590, 1.513560, 1.523520, 1.533490, 1.543450,
1.553420, 1.563380, 1.573350, 1.583310, 1.593270, 1.603240, 1.613200,
1.623160, 1.633130, 1.643090, 1.653050, 1.663010, 1.672970, 1.682930,
1.692900, 1.702860, 1.712820, 1.722780, 1.732740, 1.742700, 1.752650,
1.762610, 1.772570, 1.782530, 1.792490, 1.802450, 1.812400, 1.822360,
1.832320, 1.842270, 1.852230, 1.862190, 1.872140, .1882100, 1.880310,
1.890370, 1.900420, 1.910470, 1.920520, 1.930570, 1.940620, 1.950660,
1.960700, 1.970740, 1.980770, 1.990800, 2.000830, 2.010860, 2.020880,
2.030900, 2.040920, 2.050940, 2.060950, 2.070960, .2080970, 2.090970,
2.100980, 2.110980, 2.120970, 2.130970, 2.140960, 2.150950, 2.160940,
```

```
2.170920, 2.180900, 2.190880, 2.200860, 2.210830, 2.220810, 2.230770,
2.240740, 2.250700, 2.260660, 2.270620, 2.280580, 2.290530, 2.300480,
2.310430, 2.320370, 2.330320, 2.340260, 3.250190, 2.360130, 2.370060,
2.379990, 2.389920, 2.399840, 2.409760, 2.419680, 2.429600, 2.439510,
2.449420, 2.459330, 2.469240, 2.479140, 2.489040, 2.498940, 2.508830)

  BAND_BIN_WIDTH = (
0.015450, 0.011530, 0.011380, 0.011230, 0.011090, 0.010960, 0.010830,
0.010710, 0.010590, 0.010490, 0.010380, 0.010290, 0.010200, 0.010120,
0.010040, 0.009970, 0.009910, 0.009850, 0.009800, 0.009760, 0.009720,
0.009690, 0.009660, 0.009640, 0.009630, 0.009630, 0.009630, 0.009640,
0.009650, 0.009670, 0.009700, 0.012670, 0.010880, 0.009560, 0.009520,
0.009500, 0.009480, 0.009470, 0.009470, 0.009470, 0.009490, 0.009510,
0.009540, 0.009580, 0.009620, 0.009680, 0.009740, 0.009810, 0.009890,
0.009970, 0.010070, 0.010170, 0.010280, 0.010390, 0.010520, 0.010650,
0.010790, 0.010940, 0.011100, 0.011260, 0.011440, 0.010160, 0.009210,
0.009790, 0.009440, 0.009440, 0.009430, 0.009420, 0.009410, 0.009410,
0.009400, 0.009400, 0.009390, 0.009390, 0.009380, 0.009380, 0.009380,
0.009380, 0.009380, 0.009380, 0.009380, 0.009380, 0.009380, 0.009380,
0.009390, 0.009390, 0.009390, 0.009400, 0.009410, 0.009410, 0.009420,
0.009430, 0.009430, 0.009440, 0.009450, 0.010090, 0.011130, 0.011140,
0.011150, 0.011150, 0.011160, 0.011160, 0.011170, 0.011170, 0.011180,
0.011180, 0.011180, 0.011190, 0.011190, 0.011190, 0.011190, 0.011190,
0.011200, 0.011200, 0.011200, 0.011200, 0.011200, 0.011190, 0.011190,
0.011190, 0.011190, 0.011190, 0.011180, 0.011180, 0.011180, 0.011170,
0.011170, 0.011160, 0.011160, 0.011150, 0.011140, 0.011140, 0.011130,
0.011120, 0.011110, 0.011110, 0.011100, 0.011090, 0.011080, 0.011070,
0.011060, 0.011050, 0.011040, 0.011030, 0.011010, 0.011000, 0.010990,
0.010980, 0.010960, 0.010950, 0.010930, 0.010920, 0.010910, 0.010890,
0.010870, 0.010860, 0.010840, 0.010820, 0.010810, 0.010790, 0.009980,
0.009970, 0.009950, 0.009940, 0.009930, 0.009910, 0.009900, 0.009890,
0.009880, 0.009860, 0.009850, 0.009840, 0.009820, 0.009810, 0.009800,
0.009790, 0.009770, 0.009760, 0.009750, 0.009730, 0.009720, 0.009710,
0.009700, 0.009680, 0.009670, 0.009660, 0.009650, 0.009630, 0.009620,
0.009610, 0.009600, 0.009580, 0.009570, 0.009560, 0.009550, 0.009530,
0.009520, 0.009510, 0.009500, 0.009490, 0.009470, 0.009460, 0.009450,
0.009440, 0.009420, 0.009410, 0.009400, 0.009390, 0.009380, 0.009360,
0.009350, 0.009340, 0.009330, 0.009320, 0.009300, 0.009290, 0.009280,
0.009270, 0.009260, 0.009250, 0.009230, 0.009220, 0.009210, 0.009200)

END_GROUP = BAND_BIN
```

A.25.6 Note on Using PDS SPECTRAL_QUBEs with ISIS Software

The Integrated Software for Imagers and Spectrometers (ISIS) system, developed by the U.S. Geological Survey, uses image qubes as its principal data structure. The PDS SPECTRAL_QUBE may be designed in such a way as to be suitable for use with ISIS. The optional keyword ISIS_STRUCTURE_VERSION is used to indicate that the SPECTRAL_QUBE is to be used with ISIS. As of this writing, “2.1” is the only valid ISIS version that can be used for this keyword:

ISIS_STRUCTURE_VERSION = “2.1”

This indicates that the PDS SPECTRAL_QUBE can be used with ISIS software version 2.1.

For data providers interested in producing PDS SPECTRAL_QUBEs with a physical data structure compatible with ISIS, consider the following. In order for a SPECTRAL_QUBE object to conform to the ISIS structure, the following are specifically required in addition to all other PDS SPECTRAL_QUBE requirements:

- Record lengths must be 512, i.e., RECORD_BYTES = 512.
- Core pixels of type UNSIGNED_INTEGER must be a single byte value, i.e., if CORE_ITEM_TYPE = UNSIGNED_INTEGER, then CORE_ITEM_BYTES = 1.
- Core pixels of type MSB_UNSIGNED_INTEGER, LSB_UNSIGNED_INTEGER, INTEGER, MSB_INTEGER, or LSB_INTEGER must be a 2-byte value, i.e., if CORE_ITEM_TYPE is one of these integer types, then CORE_ITEM_BYTES = 2.
- Suffix regions (if present) must allocate storage for 4.byte pixels.

Note: Conformance to these criteria ensures ISIS physical structure compatibility only. A fully compliant ISIS label is generated within ISIS at the time of ISIS ingestion. Existing ISIS ingestion software may need modifications to ingest specific PDS SPECTRAL_QUBEs, even when the SPECTRAL_QUBE is physically structured for ISIS.

A.25.7 Example label for a PDS SPECTRAL_QUBE intended for use with ISIS software

```
PDS_VERSION_ID                  = PDS3

/* File Identification and Structure */

RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 512
FILE_RECORDS                    = 9650

/* Pointer to Data Object */

^SPECTRAL_QUBE                  = "SAMPLE2.QUB"

/* Identification Data Elements */

DATA_SET_ID                     =
PRODUCT_ID                      =
INSTRUMENT_HOST_NAME            =
INSTRUMENT_NAME                 =
TARGET_NAME                     =
START_TIME                      =
STOP_TIME                       =
```

```
SPACECRAFT_CLOCK_START_COUNT    =
SPACECRAFT_CLOCK_STOP_COUNT     =
PRODUCT_CREATION_TIME           =

/* SPECTRAL_QUBE Object Description */

OBJECT                          = SPECTRAL_QUBE

  AXES                          = 3
  AXIS_NAME                     = (SAMPLE, LINE, BAND)
  ISIS_STRUCTURE_VERSION        = "2.1"
  MD5_CHECKSUM                  = cf65a98aff4232f5ac5171406590a929

  /* Core Description */

  CORE_ITEMS                    = (320, 272, 3)
  CORE_NAME                     = "CALIBRATED SPECTRAL RADIANCE"
  CORE_ITEM_BYTES               = 2
  CORE_ITEM_TYPE                = MSB_INTEGER
  CORE_BASE                     = 0.000000
  CORE_MULTIPLIER               = 1.000000
  CORE_UNIT                     = "WATT*CM**-2*SR**-1*UM**-1"
  CORE_NULL                     = -32768
  CORE_VALID_MINIMUM            = -32752
  CORE_LOW_REPR_SATURATION      = -32767
  CORE_LOW_INSTR_SATURATION     = -32766
  CORE_HIGH_REPR_SATURATION     = -32765
  CORE_HIGH_INSTR_SATURATION    = -32764

  /* Suffix Descriptions */

  SUFFIX_ITEMS                  = (1, 1, 2)
  SUFFIX_BYTES                  = 4

  GROUP                         = SAMPLE_SUFFIX
    SUFFIX_NAME                 = HORIZONTAL_DESTRIPE
    SUFFIX_ITEM_BYTES           = 4
    SUFFIX_ITEM_TYPE            = IEEE_REAL
    SUFFIX_BASE                 = 0.000000
    SUFFIX_MULTIPLIER           = 1.000000
    SUFFIX_VALID_MINIMUM        = 16#FFEFFFFF#
    SUFFIX_NULL                 = 16#FFFFFFFF#
    SUFFIX_LOW_REPR_SAT         = 16#FFFEFFFF#
    SUFFIX_LOW_INSTR_SAT        = 16#FFFDFFFF#
    SUFFIX_HIGH_REPR_SAT        = 16#FFFBFFFF#
    SUFFIX_HIGH_INSTR_SAT       = 16#FFFCFFFF#
  END_GROUP                     = SAMPLE_SUFFIX

  GROUP                         = LINE_SUFFIX
    SUFFIX_NAME                 = VERTICAL_DESTRIPE
    SUFFIX_ITEM_BYTES           = 4
    SUFFIX_ITEM_TYPE            = IEEE_REAL
    SUFFIX_BASE                 = 0.000000
    SUFFIX_MULTIPLIER           = 1.000000
```

```
    SUFFIX_VALID_MINIMUM        = 16#FFEFFFFF#
    SUFFIX_NULL                 = 16#FFFFFFFF#
    SUFFIX_LOW_REPR_SAT         = 16#FFFEFFFF#
    SUFFIX_LOW_INSTR_SAT        = 16#FFFDFFFF#
    SUFFIX_HIGH_REPR_SAT        = 16#FFFBFFFF#
    SUFFIX_HIGH_INSTR_SAT       = 16#FFFCFFFF#
  END_GROUP                     = LINE_SUFFIX

  GROUP                         = BAND_SUFFIX
    SUFFIX_NAME                 = (LATITUDE, LONGITUDE)
    SUFFIX_UNIT                 = (DEGREE, DEGREE)
    SUFFIX_ITEM_BYTES           = (4, 4)
    SUFFIX_ITEM_TYPE            = (IEEE_REAL, IEEE_REAL)
    SUFFIX_BASE                 = (0.000000, 0.000000)
    SUFFIX_MULTIPLIER           = (1.000000, 1.000000)
  END_GROUP                     = BAND_SUFFIX

  /* Band bin information */

  GROUP                         = BAND_BIN
    BANDS                       = 3
    BAND_BIN_UNIT               = MICROMETER
    BAND_BIN_FILTER_NUMBER      = (1, 2, 3)
    BAND_BIN_BAND_NUMBER        = (2, 3, 4)
    BAND_BIN_CENTER             = (6.78, 9.35, 14.88)
    BAND_BIN_WIDTH              = (1.01, 1.20, 0.87)
    BAND_BIN_BASE               = (0.0, 0.0, 0.0)
    BAND_BIN_MULTIPLIER         = (1.0, 1.0, 1.0)
  END_GROUP                     = BAND_BIN

  /* Map projection information */

  OBJECT                        = IMAGE_MAP_PROJECTION
    A_AXIS_RADIUS               = 1737.4000000
    B_AXIS_RADIUS               = 1737.4000000
    C_AXIS_RADIUS               = 1737.4000000
    POSITIVE_LONGITUDE_DIRECTION = EAST
    MAP_PROJECTION_TYPE         = "SINUSOIDAL EQUAL AREA"
    MAP_SCALE                   = 0.1000000
    MAP_RESOLUTION              = 303.2334900
    EASTERNMOST_LONGITUDE       = 126.0177002
    WESTERNMOST_LONGITUDE       = 120.0000000
    MINIMUM_LATITUDE            = 20.9867992
    MAXIMUM_LATITUDE            = 28.0000000
    CENTER_LONGITUDE            = 135.0000000
    REFERENCE_LATITUDE          = 0.0000000
    REFERENCE_LONGITUDE         = 0.0000000
    MAP_PROJECTION_ROTATION     = 0.0000000
    LINE_PROJECTION_OFFSET      = -8490.0381188
    SAMPLE_PROJECTION_OFFSET    = -4246.2684059
  END_OBJECT                    = IMAGE_MAP_PROJECTION

END_OBJECT                      = SPECTRAL_QUBE
END
```

A.26 SPECTRUM

The SPECTRUM object is a form of TABLE used for storing spectral measurements. The SPECTRUM object is assumed to have a number of measurements of the observation target taken in different spectral bands. The SPECTRUM object uses the same physical format specification as the TABLE object, but includes sampling parameter definitions which indicate the spectral region measured in successive COLUMNs or ROWs. The common sampling parameters for SPECTRUM objects are wavelength, frequency, or velocity.

A regularly sampled SPECTRUM can be stored either horizontally as a one-row table with a single column containing *n* samples (indicated in the COLUMN definition by "ITEMS = *n*"), or vertically as a one-column table with *n* rows where each row contains a sample of the spectrum. The vertical format allows additional columns to be defined for related parameters for each sample value (e.g., error bars). These related columns may also be described in a separate PREFIX or SUFFIX table.

In the horizontal format, the sampling parameter specifications are included in the COLUMN definition. For a vertically defined SPECTRUM, the sampling parameter information is provided in the SPECTRUM object, since it is describing the spectral variation between the rows of the data. An irregularly sampled SPECTRUM must be stored horizontally, with each specific spectral range identified as a separate column.

A.26.1 Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES

A.26.2 Optional Keywords

1. NAME
2. SAMPLING_PARAMETER_NAME
3. SAMPLING_PARAMETER_UNIT
4. SAMPLING_PARAMETER_INTERVAL
5. ROW_PREFIX_BYTES
6. ROW_SUFFIX_BYTES
7. MINIMUM_SAMPLING_PARAMETER
8. MAXIMUM_SAMPLING_PARAMETER
9. DERIVED_MINIMUM
10. DERIVED_MAXIMUM
11. DESCRIPTION

A.26.3 Required Objects

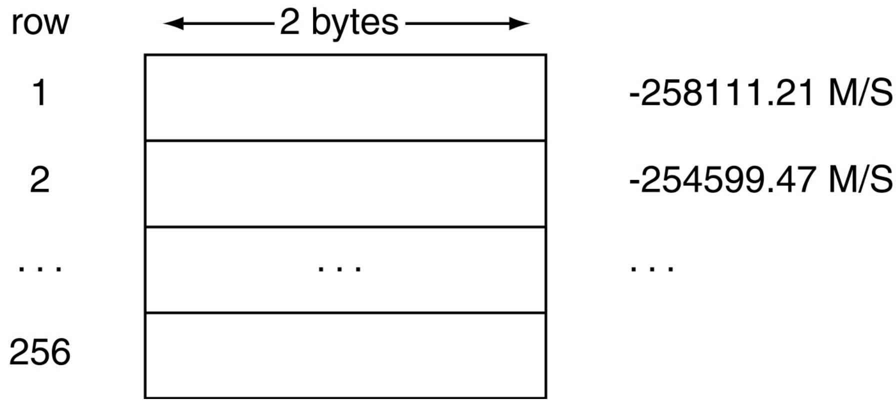
1. COLUMN

A.26.4 Optional Objects

1. CONTAINER

A.26.5 Example

This example illustrates a SPECTRUM data object stored in a vertical format. The data are regularly sampled at intervals of 99.09618 meters/second and data samples are stored in successive ROWS.




```
PDS_VERSION_ID                 = PDS3
RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 2
FILE_RECORDS                   = 256
PRODUCT_ID                     = "RSSL007.DAT"
DATA_SET_ID                    = "IHW-C-RSSL-3-EDR-HALLEY-V1.0"
TARGET_NAME                    = "HALLEY"
INSTRUMENT_HOST_NAME           = "IHW RADIO STUDIES NETWORK"
INSTRUMENT_NAME                = "RADIO SPECTRAL LINE DATA"
OBSERVATION_ID                 = "621270"
START_TIME                     = 1985-11-10T00:43:12.000
STOP_TIME                      = 1985-11-10T00:43:12.000
PRODUCT_CREATION_TIME          = "UNK"

/* Record Pointer to Major Object */
^TOTAL_INTENSITY_SPECTRUM      = "RSSL0007.DAT"

/* Object Description */

OBJECT                         = SPECTRUM
  INTERCHANGE_FORMAT           = BINARY
```

```
    ROWS                         = 256
    ROW_BYTES                    = 2
    COLUMNS                      = 1
    SAMPLING_PARAMETER_NAME      = "VELO_COM"
    MINIMUM_SAMPLING_PARAMETER   = -1.268431E+04
    SAMPLING_PARAMETER_INTERVAL  = 9.909618E+01
    SAMPLING_PARAMETER_UNIT      = "METERS/SECOND"
    DESCRIPTION                  = "Radio Studies; Spectral Line intensity
                                    spectrum.  Spectrum is organized as 1
                                    column with 256 rows.  Each row
                                    contains a spectral value for the
                                    velocity derived from the sampling
                                    parameter information associated with
                                    each row."

    OBJECT                       = COLUMN
      NAME                       = FLUX_DENSITY
      DATA_TYPE                  = MSB_INTEGER
      START_BYTE                 = 1
      BYTES                      = 2
      SCALING_FACTOR             = 7.251200E-04
      OFFSET                     = 0.000000E+01
      DERIVED_MINIMUM            = 2.380000E+01
      DERIVED_MAXIMUM            = 3.490000E+01
    END_OBJECT                   = COLUMN
  END_OBJECT                     = SPECTRUM

  END
```

A.27 SPICE KERNEL

The SPICE_KERNEL object describes a single kernel file in a collection of SPICE kernels. SPICE kernels provide ancillary data needed to support the planning and subsequent analysis of space science observations. The SPICE system includes the software and documentation required to read the SPICE Kernels and use the data contained therein to help plan observations or interpret space science data. This software and associated documentation are collectively called the NAIF Toolkit.

Kernel files are the major components of the SPICE system. Each type of kernel, indicated by the KERNEL_TYPE keyword, corresponds to one of these components and has a specific abbreviation. The major kernel types, their abbreviations, and the associated file extension(s) are listed in the following table. (For a complete list of file extensions, see Section 10.2.3.)

KERNEL_TYPE	**Abbreviation**	**File Extension**	**Contents**
EPHEMERIS	SPK	.BSP – *binary* .XSP – *transfer*	Spacecraft, planet, satellite, or other target body epehemeris data to provide position and velocity of a target as a function of time
TARGET_CONSTANTS	PCK	.TPC	Cartographic constants for a planet, satellite, comet, or asteroid
INSTRUMENT	IK	.TI	Collected science instrument information, including dpecification of the mounting alignment, internal timing, and other information needed to interpret measurements made with a particular instrument
POINTING	CK	.BC – *binary* .XC – *transfer*	Pointing data, e.g., the inertially referenced attitude for a spacecraft structure upon which instruments are mounted, given as a function of time
EVENTS	EK	.XES	Event information, e.g., spacecraft and instrument commands, ground data system event logs, and experimenter's notebook comments
LEAPSECONDS	LSK	.TLS	An account of the leapseconds needed to correlate civil time (UTC) to ephemeris time (TDB), the measure of time used in the SP kernel files
SPACECRAFT_CLOCK-_COEFFICIENTS	SCLK	.TSC	Data needed to correlate a spacecraft clock to ephemeris time
FRAMES	FK	.TF	Frame information, used to define reference frames used within SPICE.

Data products referencing a particular SPICE kernel do so by including the SOURCE_PRODUCT_ID keyword in their label with a value corresponding to that of the PRODUCT_ID keyword in the SPICE_KERNEL label. (The PRODUCT_ID keyword is unique to a data product.)

A.27.1 Required Keywords

1. DESCRIPTION
2. INTERCHANGE_FORMAT
3. KERNEL_TYPE

A.27.2 Optional Keywords

Any

A.27.3 Required Objects

None

A.27.4 Optional Objects

None

A.27.5 Example

Following is an example of a SPICE CK (pointing) kernel label. This label would be attached to the CK file, and thus would be immediately followed by the internal CK file header. (This example was fabricated for use here based on existing examples.)

```
PDS_VERSION_ID                 = PDS3
RECORD_TYPE                    = STREAM
MISSION_NAME                   = MARS_OBSERVER
SPACECRAFT_NAME                = MARS_OBSERVER
DATA_SET_ID                    = "MO-M-SPICE-6-CK-V1.0"
FILE_NAME                      = "NAF0000D.TC"
PRODUCT_ID                     = "NAF0000D-CK"
PRODUCT_CREATION_TIME          = 1992-04-14T12:00:00
PRODUCER_ID                    = "NAIF"
MISSION_PHASE_TYPE             = "ORBIT"
PRODUCT_VERSION_TYPE           = "TEST"
START_TIME                     = 1994-01-06T00:00:00
STOP_TIME                      = 1994-02-04T23:55:00
SPACECRAFT_CLOCK_START_COUNT   = "3/76681108.213"
SPACECRAFT_CLOCK_STOP_COUNT    = "4/79373491.118"
TARGET_NAME                    = MARS
INSTRUMENT_NAME                = "MARS OBSERVER SPACECRAFT"
INSTRUMENT_ID                  = MO
SOURCE_PRODUCT_ID              =
    {"NAF0000C.BSP","NAF0000C.TLS","NAF0000C.TSC"}
NOTE                           = "BASED ON EPHEMERIS IN NAF0000C.BSP.
    FOR SOFTWARE TESTING ONLY."

OBJECT                         = SPICE_KERNEL
  INTERCHANGE_FORMAT           = ASCII
```

```
    KERNEL_TYPE                   = POINTING
    DESCRIPTION                   = "This is a SPICE kernel file, designed
      to be accessed using NAIF Toolkit software. Contact your flight
      project representative or the NAIF node of the Planetary Data System
      if you wish to obtain a copy of the NAIF Toolkit.  The Toolkit
      consists of portable FORTRAN 77 code and extensive user
      documentation."
  END_OBJECT                      = SPICE_KERNEL
  END
```

A.28 SPREADSHEET

The SPREADSHEET is a natural storage format for data products in which the data rows are sparsely populated or field values have variable lengths.

A SPREADSHEET definition describes a collection of logically uniform rows containing ASCII values stored in variable-width fields separated by field delimiters. Each row within a SPREADSHEET has the same number of fields, in the same field order; and each field contains the same logical content. By definition, the SPREADSHEET object is used only to describe ASCII data objects. Therefore, it is not necessary to include the INTERCHANGE_FORMAT keyword within the object keyword list. The rows and fields of the SPREADSHEET object provide a natural correspondence to the rows and columns of fixed-width tables. Each field is defined by a variable width FIELD object (see section A.14); the value of the FIELDS keyword is the total number of FIELD objects defined in the SPREADSHEET. All SPREADSHEET objects have variable-length records and have rows delimited by carriage-return line-feed (<CR><LF>) ASCII line termination characters.

A.28.1 Required Keywords

4. ROWS
5. ROW_BYTES
6. FIELDS
7. FIELD_DELIMITER

A.28.2 Optional Keywords

10. NAME
11. DESCRIPTION
12. PSDD

A.28.3 Required Objects

1. FIELD

A.28.4 Optional Objects

None

Notes:

1. The RECORD_BYTES keyword in the implied file object definition of the PDS label containing a SPREADSHEET object definition should specify the actual number of bytes in the longest record within the file being described. If the file contains several components, this longest record may not necessarily be in the SPREADSHEET.

2. The ROW_BYTES keyword within the SPREADSHEET object definition is used to specify the maximum number of bytes that could be contained in a row in the SPREADSHEET object (i.e. the sum of all the FIELD object BYTES values, plus the number of delimiters and quotation marks, plus the 2 bytes for the <CR><LF> line termination).

A.28.5 Required SPREADSHEET Formats

The SPREADSHEET is an ASCII data object. Its records contain fixed numbers of variable-length fields and are delimited by carriage-return line-feed pairs. The FIELD delimiter can be COMMA, SEMICOLON, TAB, or VERTICAL_BAR; subfields (if any) are delimited by the same character.

The ASCII format makes the SPREADSHEET readable by both machines and humans. The relative loss in human readability (compared to the TABLE object) is mitigated by more efficient storage, especially for sparsely populated fields.

Several keywords take on special meanings in the SPREADSHEET context. BYTES (and ITEM_BYTES, if used) gives the maximum allowable number of bytes in the FIELD (ITEM). ROW_BYTES is the maximum allowable number of bytes in the row, including delimiters, quotation marks, and the carriage-return line-feed pair. RECORD_TYPE within the implied parent file object is always STREAM. RECORD_BYTES within the implied file is the actual number of bytes in the longest record, including the carriage-return line-feed pair. If the file contains more than the SPREADSHEET, however, the longest record may not be a SPREADSHEET record.

A.28.6 Recommended SPREADSHEET Formats

The recommended format for SPREADSHEET objects is a comma-separated value format in which string fields are enclosed in double quotes. This format can be imported directly into many commercial data management systems and spreadsheet applications.

The recommended file name extension for files containing SPREADSHEET objects is CSV (e.g., MYDATA.CSV), but the CSV extension does not necessarily imply that the field delimiter is COMMA.

Example - Recommended SPREADSHEET

The following example shows a sparse matrix described as a SPREADSHEET object. The longest record is 85-bytes. Note that delimiters (double quotes and commas) and line terminators (<CR><LF>) are included in the byte count for each record (RECORD_BYTES) and row (ROW_BYTES).

Contents of file "MYDATA.CSV":

```
2004-03-04T00:00:00.012,0.45,"MODE 1",0,,,1,,,-1,12,5,1,2,1,1,0,1,3,1,0<CR><LF>
2004-03-04T00:00:01.012,0.45,"MODE 1",1,,,1,,,6,9,15,8,7,2,1,1,0,0,1,0<CR><LF>
2004-03-04T00:00:02.012,0.45,"MODE 1",2,,,5,,,25,15,10,4,2,1,1,1,1,0,1,1<CR><LF>
2004-03-04T00:00:03.012,0.45,"MODE 1",1,,,1,,,2,4,8,3,1,1,1,1,1,0,0<CR><LF>
2004-03-04T00:00:04.012,0.45,"MODE 5",1,1,3,1,1,2,3,1,1,2,2,1,4,3,1,1,4,1,1,0<CR><LF>
2004-03-04T00:00:05.012,0.45,"MODE 5",1,5,4,2,1,1,1,1,2,0,0,1,0,1,1,0,0,0,0,0<CR><LF>
2004-03-04T00:00:06.012,0.45,"MODE 5",1,6,3,5,4,3,1,,0,1,1,1,1,2,1,1,1,3,1,0<CR><LF>
2004-03-04T00:00:07.012,0.45,"MODE 6",,,,3,,,5,,1,,1,3,,2,3,,,,<CR><LF>
2004-03-04T00:00:08.012,0.45,"MODE 6",,,,,1,,,2,,1,,1,4,,1,2,,,,<CR><LF>
2004-03-04T00:00:09.012,0.45,"MODE 6",,,,1,,,,,1,1,1,,,1,,,,,<CR><LF>
2004-03-04T00:00:10.017,4.00,"MODE 11",,,,,8,15,14,21,24,18,15,10,8,9,11,6,-1,9,8,6<CR><LF>
2004-03-04T00:00:15.017,4.00,"MODE 11",,,,,8,12,17,35,20,12,5,1,2,1,1,8,11,7,8,6<CR><LF>
2004-03-04T00:00:20.017,4.00,"MODE 11",,,,,4,8,12,32,24,12,15,4,3,1,1,6,7,3,5,2<CR><LF>
2004-03-04T00:00:25.017,4.00,"MODE 13",,,,,1,5,12,12,14,12,5,1,1,7,2,4,,,,<CR><LF>
2004-03-04T00:00:30.017,4.00,"MODE 13",,,,,1,5,5,14,16,10,8,3,1,5,3,2,,,,<CR><LF>
2004-03-04T00:00:35.017,4.00,"MODE 13",,,,,1,2,3,2,19,43,21,17,4,8,3,1,,,,<CR><LF>
2004-03-04T00:00:40.017,4.00,"MODE 13",,,,,1,2,1,2,4,12,9,3,1,1,1,1,,,,<CR><LF>
2004-03-04T00:00:45.017,4.00,"MODE 13",,,,,1,3,1,-1,9,16,7,1,1,1,1,2,,,,<CR><LF>
2004-03-04T00:00:50.017,4.00,"MODE 13",,,,,1,2,1,2,4,12,5,1,1,1,1,1,,,,<CR><LF>
2004-03-04T00:00:55.017,4.00,"MODE 13",,,,,1,2,1,2,4,10,5,1,1,1,1,1,,,,<CR><LF>
```

```
MYDATA.CSV is an example data file described by a SPREADSHEET object
definition within a PDS label. The longest record in this file is 85 bytes
(record 11) and this value is assigned to the RECORD_BYTES keyword.  However,
records described by this SPREADSHEET definition could be as long as 163
bytes (see example label below).  The value assigned to the ROW_BYTES keyword
(163) is the maximum possible row size (bytes) described by the SPREADSHEET
object definition.

        Bytes     Field
          23    - Time (23)
           8    - delimiter + duration (7)
          10    - delimiter + quotes(2) + mode string (7)
          60    - delimiter + electrons (59)
          60    - delimiter + ions (59)
       +   2    - CR + LF
      -----------------------
       = 163    = ROW_BYTES


Contents of file "MYDATA.LBL":

PDS_VERSION_ID          = PDS3
RECORD_TYPE             = STREAM
RECORD_BYTES            = 85      /* Largest actual record in the file */
FILE_RECORDS            = 20
^SPREADSHEET            = "MYDATA.CSV"
DATA_SET_ID             = "CO-S-INST-2-DUMMY-DATA-V1.0"
SPACECRAFT_NAME         = "CASSINI ORBITER"
INSTRUMENT_NAME         = "MY INSTRUMENT"
```

```
TARGET_NAME             = {"SATURN", "SOLAR_WIND"}
PRODUCT_ID              = "MYDATA.CSV"
PRODUCT_CREATION_TIME   = 2004-08-04T11:15:00
START_TIME              = 2004-03-04T00:00:00.012
STOP_TIME               = 2004-03-04T00:00:55.017
DESCRIPTION             = "This file contains an example
                           sparse matrix data object (SPREADSHEET)."

OBJECT                  = SPREADSHEET
  ROWS                  = 20
  ROW_BYTES             = 163     /* Size of longest possible row*/
  FIELDS                = 5
  FIELD_DELIMITER       = "COMMA"

  OBJECT                = FIELD
     NAME               = "TIME"
     DATA_TYPE          = TIME
     FIELD_NUMBER       = 1
     BYTES              = 23
     DESCRIPTION        = "Spacecraft event time (UT) for this data record."
 END_OBJECT             = FIELD

  OBJECT                = FIELD
    NAME                = "DURATION"
    FIELD_NUMBER        = 2
    BYTES               = 7
    FORMAT              = "F7.2"
    DATA_TYPE           = "ASCII_REAL"
    UNITS               = "SECOND"
    DESCRIPTION         = "Time interval over which counting was performed
(seconds)."
  END_OBJECT            = FIELD

  OBJECT                = FIELD
    NAME                = "MODE"
    FIELD_NUMBER        = 3
    BYTES               = 7 /* doesn't count bytes occupied by double
quotes*/
    FORMAT              = "A7"
    DATA_TYPE           = "CHARACTER"
    DESCRIPTION         = "Scan mode name. See the instrument description for
                           a complete list of scan mode names and
      properties."
  END_OBJECT            = FIELD

  OBJECT                = FIELD
    NAME                = "ELECTRON COUNTS"
    FIELD_NUMBER        = 4
    BYTES               = 59    /* Maximum bytes including item delimiters */
    ITEMS               = 10
    ITEM_BYTES          = 5     /* Maximum item bytes */
    FORMAT              = "I5"
    DATA_TYPE           = "ASCII_INTEGER"
    UNITS               = "COUNTS"
    MISSING_CONSTANT    = -1
    DESCRIPTION         = "This field contains electron counts from channels
                           E1-E10. Items without values indicate channels not
                           counted during the interval. Values of zero denote
```

```
                              counted channels in which no electrons were
                              detected. Values of -1 denote corrupted data,
                              excluded from the data file (counted, but value
                              undefined)."
  END_OBJECT                = FIELD

  OBJECT                    = FIELD
     NAME                   = "ION COUNTS"
     FIELD_NUMBER           = 5    /* 5th FIELD object in label */
     BYTES                  = 59
     ITEMS                  = 10
     ITEM_BYTES             = 5
     FORMAT                 = "I5"
     DATA_TYPE              = "ASCII_INTEGER"
     UNITS                  = "COUNTS"
     MISSING_CONSTANT       = -1
     DESCRIPTION            = "This field contains ion counts from channels D1-
                              D10. Items without values indicate channels not
                              counted during the interval.  Values of zero
                              denote counted channels in which no ions were
                              detected.  Values of -1 denote corrupted data,
                              excluded from the data file (counted, but value
                              undefined)."
  END_OBJECT                = FIELD
END_OBJECT                  = SPREADSHEET
END
```

A.29 TABLE

TABLEs are a natural storage format for collections of data from many instruments. They are often the most effective way of storing much of the meta-data used to identify and describe instrument observations.

The TABLE object is a uniform collection of rows containing ASCII or binary values stored in columns. The INTERCHANGE_FORMAT keyword is used to distinguish between TABLEs containing only ASCII columns and those containing binary data. The rows and columns of the TABLE object provide a natural correspondence to the records and fields often defined in interface specifications for existing data products. Each field is defined as a fixed-width COLUMN object; the value of the COLUMNS keyword is the total number of COLUMN objects defined in the label. All TABLE objects must have fixed-width records.

Many variations on the basic TABLE object are possible with the addition of optional keywords and/or objects. While it is possible to create very complex row structures, these are often not the best choices for archival data products. Recommended ASCII and binary table formats are described and illustrated below.

A.29.1 Keywords

A.29.1.1 Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES

A.29.1.2 Optional Keywords

1. NAME
2. DESCRIPTION
3. ROW_PREFIX_BYTES
4. ROW_SUFFIX_BYTES
5. TABLE_STORAGE_TYPE

A.29.1.3 Required Objects

1. COLUMN

A.29.1.4 Optional Objects

1. CONTAINER

A.29.2 ASCII vs. BINARY formats

ASCII tables provide the most portable format for access across a wide variety of computer platforms. They are also easily imported into a number of database management systems and spreadsheet applications. For these reasons, the PDS recommends the use of ASCII table formats whenever possible for archive products.

ASCII formats are generally less efficient for storing large quantities of numeric data. In addition, raw or minimally processed data products and many pre-existing data products undergoing restoration are only available in binary formats.Where conversion to an ASCII format is not cost effective or is otherwise undesirable, BINARY table formats may be used.

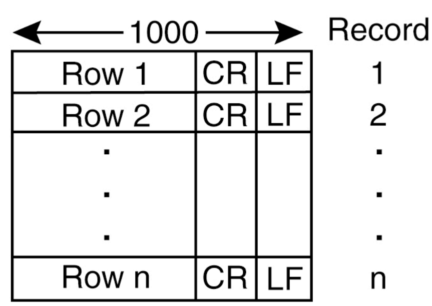
A.29.3 Recommended ASCII TABLE Format

The recommended format for ASCII TABLE files is a comma-separated value format in which the string fields are enclosed in double quotes. ASCII tables must have fixed-length records and should use carriage-return/linefeed (<CR><LF>) delimiters. Numeric fields are right-justified in the allotted space and character fields are left-justified and blank padded on the right. This table format can be imported directly into many commercial data management systems.

The field delimiters and quotation marks must occur between the defined COLUMNs. That is, the START_BYTE for a string column should not point to the opening quotation mark, but the first character in the field itself. Similarly, the BYTES values for the columns should not include the commas at the end of the values. For example, a twelve character COLUMN called SPACECRAFT_NAME would be represented in the table as `"VOYAGER 1   "` rather than `"   VOYAGER 1"` or `"VOYAGER 1"`.

The following label fragment illustrates the general characteristics of the recommended ASCII TABLE format for a table with 1000-byte records:

```
RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 1000
...
OBJECT                      = TABLE
  INTERCHANGE_FORMAT        = ASCII
  ROW_BYTES                 = 1000
  ...
END_OBJECT                  = TABLE
```



A.29.3.1 Example - Recommended ASCII TABLE

The following example is an ASCII index table with 71-byte records. Note that for ASCII tables, the delimiters (double quotes and commas) and line terminators (<CR><LF>) *are* included in the byte count for each record (RECORD_BYTES). In this example, the delimiters are also included in the byte count for each row (ROW_BYTES). The <CR><LF> characters have been placed in columns 70 and 71.

> **Note:** The example following is an INDEX_TABLE, a specific type of (ASCII) TABLE object. Two rows of numbers indicating the byte count (read vertically) have been added above the data file contents to facilitate comparison with the label. These rows would *not* appear in the actual data file.

Contents of file "INDEX.TAB":

```
000000000111111111122222222223333333333444444444455555555556666666666 7   7
123456789012345678901234567890123456789012345678901234567890123456789 0   1
"F-MIDR ","F-MIDR.40N286;1 ","C", 42, 37,289,282,"F40N286/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.20N280;1 ","C", 22, 17,283,277,"F20N280/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.20N286;1 ","C", 22, 17,289,283,"F20N286/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.00N279;1 ","R",  2, -2,281,275,"F00N279/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.05N290;1 ","C",  7,  2,292,286,"F05N290/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.05S279;1 ","R", -2, -7,281,275,"F05S279/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.10S284;1 ","C", -7,-12,287,281,"F10S284/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.10S290;1 ","R", -7,-12,292,286,"F10S290/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.15S283;1 ","R",-12,-17,286,279,"F15S283/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.15S289;1 ","R",-12,-17,291,285,"F15S289/FRAME.LBL "<CR><LF>
```

Contents of file "INDEX.LBL":

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 71
FILE_RECORDS                  = 10
^INDEX_TABLE                  = "INDEX.TAB"

DATA_SET_ID                   = "MGN-V-RDRS-5-MIDR-FULL-RES-V1.0"
VOLUME_ID                     = MG_7777
PRODUCT_ID                    = "FMIDR.XYZ"
SPACECRAFT_NAME               = MAGELLAN
INSTRUMENT_NAME               = "RADAR SYSTEM"
TARGET_NAME                   = VENUS
PRODUCT_CREATION_TIME         = 1999-02-23t11:15:07
MISSION_PHASE_NAME            = PRIMARY_MISSION
NOTE                          = "This table lists all MIDRs on this
   volume.  It also includes the latitude and longitude range for each
   MIDR and the directory in which it is found."
```

```
OBJECT                          = INDEX_TABLE
  INTERCHANGE_FORMAT            = ASCII
  ROWS                          = 10
  COLUMNS                       = 8
  ROW_BYTES                     = 71
  INDEX_TYPE                    = SINGLE

  OBJECT                        = COLUMN
    NAME                        = PRODUCT_TYPE
    DESCRIPTION                 = "Magellan DMAT type code.  Possible
                                   values are F-MIDR, C1-MIDR, C2-MIDR,
                                   C3-MIDR, and P-MIDR."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 2
    BYTES                       = 7
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = PRODUCT_ID
    DESCRIPTION                 = "Magellan DMAT name of product.
                                    Example:  F-MIDR.20N334;1"
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 12
    BYTES                       = 16
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = SEAM_CORRECTION_TYPE
    DESCRIPTION                 = "A value of C indicates that cross-
                                   track seam correction has been applied.
                                   A value of R indicates that the
                                   correction has not been applied."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 31
    BYTES                       = 1
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = MAXIMUM_LATITUDE
    DESCRIPTION                 = "Northernmost frame latitude rounded to
                                   the nearest degree."
    DATA_TYPE                   = INTEGER
    UNIT                        = DEGREE
    START_BYTE                  = 34
    BYTES                       = 3
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = MINIMUM_LATITUDE
    DESCRIPTION                 = "Southernmost frame latitude rounded to
                                   the nearest degree."
    DATA_TYPE                   = INTEGER
    UNIT                        = DEGREE
    START_BYTE                  = 38
```

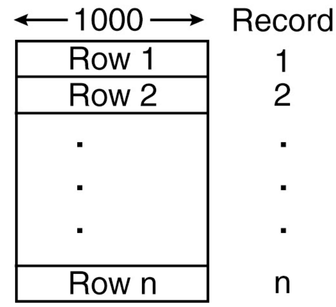
```
    BYTES                       = 3
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = EASTERNMOST_LONGITUDE
    DESCRIPTION                 = "Easternmost frame longitude rounded to
                                   the nearest degree."
    DATA_TYPE                   = INTEGER
    UNIT                        = DEGREE
    START_BYTE                  = 42
    BYTES                       = 3
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = WESTERNMOST_LONGITUDE
    DESCRIPTION                 = "Westernmost frame longitude rounded to
                                   the nearest degree."
    DATA_TYPE                   = INTEGER
    UNIT                        = DEGREE
    START_BYTE                  = 46
    BYTES                       = 3
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = FILE_SPECIFICATION_NAME
    DESCRIPTION                 = "Path and file name of frame table
                                   relative to CD-ROM root directory."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 51
    BYTES                       = 18
  END_OBJECT                    = COLUMN

END_OBJECT                      = INDEX_TABLE
END
```

A.29.4 Recommended BINARY TABLE Format

In the case of binary data, PDS recommends a format in which one data record corresponds to one row in the TABLE. Unused or spare bytes embedded within the record should be defined as COLUMNs (one for each chunk of contiguous unused bytes) named “SPARE”, both for completeness and to facilitate automated validation of the TABLE structure. For reasons of portability, BIT_COLUMN objects within COLUMNs are discouraged. Whenever possible, bit fields should be unpacked into more portable, byte-oriented COLUMNS.

The following label fragment illustrates the general characteristics of the recommended binary TABLE format for a table with 1000-byte records:

```
RECORD_TYPE                  = FIXED_LENGTH
RECORD_BYTES                 = 1000
...
OBJECT                       = TABLE
  INTERCHANGE_FORMAT         = BINARY
  ROW_BYTES                  = 1000
  ...
END_OBJECT                   = TABLE
```



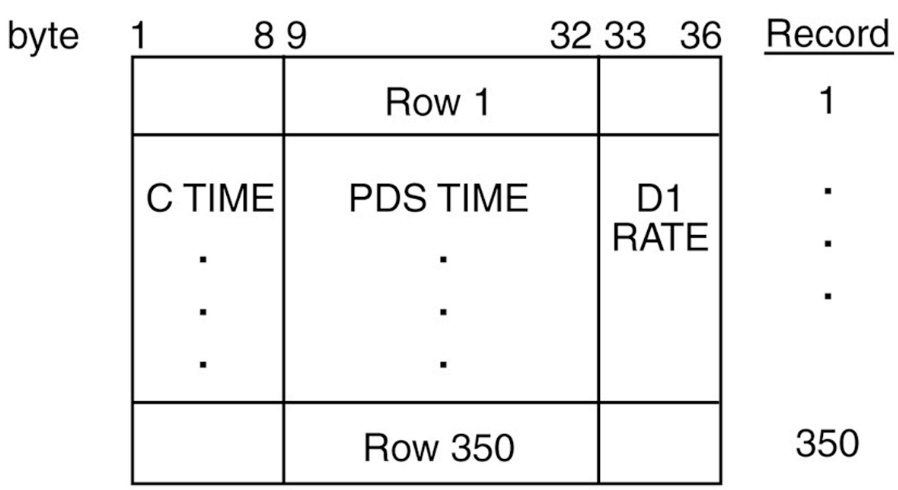
A.29.4.1 Example - Recommended Binary TABLE

Following is an example of a binary table containing three columns of data. The first two columns provide TIME information in both the PDS standard UTC format and an alternate format. The third column provides uncalibrated instrument measurements for the given time. The binary data reside in the file “T890825.DAT”. The detached label file, “T890825.LBL” providing the complete description, is presented below.

Note: The label makes use of a format file, pointed to by the ^STRUCTURE keyword in the TABLE definition, to include a set of column definitions held in an external file (“CRSDATA.FMT”). The contents of this structure file are also provided below.

This table could also be represented as a TIME_SERIES by the addition of sampling parameter keywords to describe the row-to-row variation in the table.

Contents of label file “T890825.DAT”:



Contents of label file “T890825.LBL”:

```
PDS_VERSION_ID                = PDS3

/* File Characteristic Keywords */
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 36
FILE_RECORDS                  = 350
HARDWARE_MODEL_ID             = "SUN SPARC STATION"
OPERATING_SYSTEM_ID           = "SUN OS 4.1.1"

/* Data Object Pointers */
^TABLE                        = "T890825.DAT"

/* Identification Keywords */
DATA_SET_ID                   = "VG2-N-CRS-4-SUMM-D1-96SEC-V1.0"
SPACECRAFT_NAME               = "VOYAGER 2"
INSTRUMENT_NAME               = "COSMIC RAY SYSTEM"
TARGET_NAME                   = NEPTUNE
START_TIME                    = 1989-08-25T00:00:00.000
STOP_TIME                     = 1989-08-25T09:58:02.000
MISSION_PHASE_NAME            = "NEPTUNE ENCOUNTER"
PRODUCT_ID                    = "T890825.DAT"
PRODUCT_CREATION_TIME         = "UNK"
SPACECRAFT_CLOCK_START_COUNT  = "UNK"
SPACECRAFT_CLOCK_STOP_COUNT   = "UNK"

/* Data Object Descriptions */
OBJECT                        = TABLE
  INTERCHANGE_FORMAT          = BINARY
  ROWS                        = 350
  COLUMNS                     = 3
  ROW_BYTES                   = 36
  ^STRUCTURE                  = "CRSDATA.FMT"
END_OBJECT                    = TABLE
END
```

Contents of file “CRSDATA.FMT”:

```
OBJECT                        = COLUMN
  NAME                        = "C TIME"
  UNIT                        = "SECOND"
  DATA_TYPE                   = REAL
  START_BYTE                  = 1
  BYTES                       = 8
  MISSING                     = 1.0E+32
  DESCRIPTION                 = "Time column. This field contains time
                                 in seconds after Jan 01, 1966 but is
                                 displayed in the default time format
                                 selected by the user."
END_OBJECT                    = COLUMN

OBJECT                        = COLUMN
```

```
    NAME                        = "PDS TIME"
    UNIT                        = "TIME"
    DATA_TYPE                   = TIME
    START_BYTE                  = 9
    BYTES                       = 24
    DESCRIPTION                 = "Date/Time string of the form yyyy-mm-
        ddThh:mm:ss.sss such that the representation of the date Jan 01,
        2000  00:00:00.000 would be 2000-01-01T00:00:00.000."
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = "D1 RATE"
    UNIT                        = "COUNT"
    DATA_TYPE                   = "REAL"
    START_BYTE                  = 33
    BYTES                       = 4
    MISSING                     = 1.0E+32
    DESCRIPTION                 = "The D1 rate is approximately
        porportional to the omnidirectional flux of electrons with
        kinetic energy > ~1MeV. To obtain greater accuracy, the D1
        calibration tables (see catalog) should be applied."
  END_OBJECT                    = COLUMN
```

A.29.5 TABLE Variations

This section addresses a number of variations on the basic TABLE object that arise when TABLEs appear in data files with other objects, or where file attributes may differ from the one row-one record approach recommended above. The variations discussed below are equally applicable to the other TABLE-type objects, SERIES and SPECTRUM.

This section is not intended to be a complete reference for TABLE variations. Within the following examples, some illustrate a recommended data modelling approach, some illustrate alternate approaches, and other examples are included solely to document their existence.

A.29.5.1 Record blocking in Fixed Length TABLES

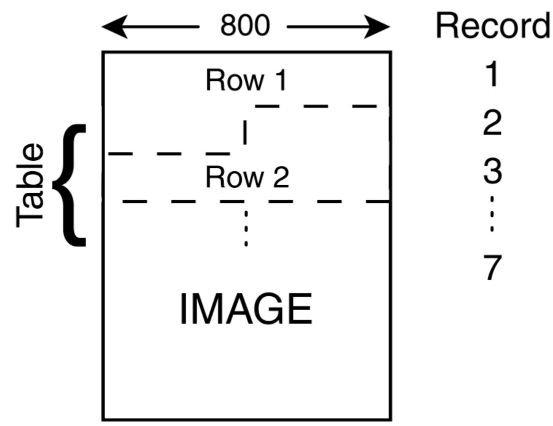
In the PDS recommended TABLE format, ROW_BYTES = RECORD_BYTES, but this is not always achievable. TABLEs are sometimes packaged with other objects in the same file, or binary data may be blocked into larger records, both resulting in cases where the TABLE row size will not match the file record width.

Rows in either ASCII or binary tables may be either larger or smaller than the physical record size specified by the RECORD_BYTES keyword. Regardless of the relationship between row size and record size, the RECORD_BYTES keyword must *always* reflect the actual physical record size, while ROW_BYTES must *always* be the logical size of one row of the TABLE object.

A.29.5.1.1 Example: Binary Table with ROW_BYTES > RECORD_BYTES

The following label fragment illustrates a case in which the record size of the file is smaller than the row size of the TABLE. Note that the table rows may straddle record boundaries. Each object, however, must begin on a record boundary, so it is possible that some padding may be required between the end of the TABLE object and the beginning of the IMAGE object, depending on the number of rows in the TABLE:

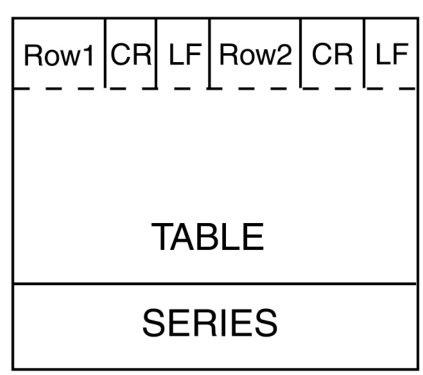
```
RECORD_TYPE          = FIXED_LENGTH
RECORD_BYTES         = 800
^TABLE               =("IMAGE.IMG",1)
^IMAGE               =("IMAGE.IMG",7)
  ...
OBJECT               = TABLE
  INTERCHANGE_FORMAT = BINARY
  ROW_BYTES          = 1200
    ...
END_OBJECT           = TABLE

OBJECT               = IMAGE
  SAMPLES            = 800
  SAMPLE_BITS        = 8
    ...
END_OBJECT           = IMAGE
```



A.29.5.1.2 Example: ASCII Table with ROW_BYTES < RECORD_BYTES

The label fragment below illustrates a case in which the row size of the TABLE is smaller than the record size of the file. It is not required that the file record size be an integral multiple of the table row size; as illustrated above, table rows may straddle record boundaries. Also as above, it is possible that some padding will be required to ensure that the subsequent SERIES object begins on a record boundary.

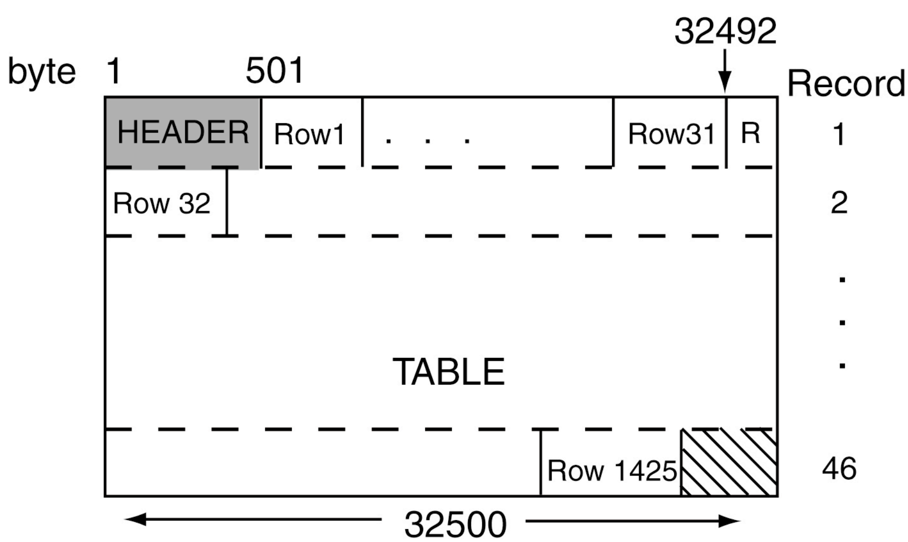
```
RECORD_TYPE            = FIXED_LENGTH
RECORD_BYTES           = 800
^TABLE                 = ("EXAMPLE.TAB",1)
^SERIES                = ("EXAMPLE.TAB",1214)
...

OBJECT                 = TABLE
  INTERCHANGE_FORMAT   = ASCII
  ROW_BYTES            = 400
  ...
END_OBJECT             = TABLE

OBJECT                 = SERIES
  INTERCHANGE_FORMAT   = ASCII
  ROW_BYTES            = 800
  ...
```



```
END_OBJECT             = SERIES
```

A.29.5.1.3 Example: Binary Table with ROW_BYTES < RECORD_BYTES

It is often the case that a data object such as a TABLE is preceeded by a header containing observational parameters or, as frequently happens with TABLEs, a set of column headings. The label below illustrates a case in which a HEADER object containing a single 500-byte row preceeds a TABLE having 1032-byte records. The file is physically blocked into records of 32,500 bytes. Note that in this case the HEADER record is *not* padded out to the full block size. Instead, a byte offset (rather than a record offset) is used to indicate the start of the TABLE object. (This example also includes COLUMN definitions contained in an external format file, a fragment of the contents of which is also shown below, following the label.)




```
PDS_VERSION_ID                = PDS3

/* FILE CHARACTERISTICS */
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 32500
FILE_RECORDS                  = 46
^HEADER                       = ("ADF01141.3",1)
^TABLE                        = ("ADF01141.3",501<BYTES>)

/* IDENTIFICATION KEYWORDS */
DATA_SET_ID                   = "MGN-V-RDRS-5-CDR-ALT/RAD-V1.0"
PRODUCT_ID                    = "ADF01141.3"
TARGET_NAME                   = VENUS
SPACECRAFT_NAME               = MAGELLAN
INSTRUMENT_NAME               = "RADAR SYSTEM"
MISSION_PHASE_NAME            = PRIMARY_MISSION
PRODUCT_CREATION_TIME         = 1991-07-23T06:16:02.000
ORBIT_NUMBER                  = 1141
START_TIME                    = UNK
```

```
STOP_TIME                       = UNK
SPACECRAFT_CLOCK_START_COUNT    = UNK
SPACECRAFT_CLOCK_STOP_COUNT     = UNK
HARDWARE_VERSION_ID             = 01
SOFTWARE_VERSION_ID             = 02
UPLOAD_ID                       = M0356N
NAVIGATION_SOLUTION_ID          = "ID = M0361-12 "
DESCRIPTION                     = "This file contains binary records
      describing, in time order, each altimeter footprint measured
      during an orbit of the Magellan radar mapper."

/* DATA OBJECT DEFINITION  DESCRIPTIONS */
OBJECT                          = HEADER
  HEADER_TYPE                   = SFDU
  BYTES                         = 500
END_OBJECT                      = HEADER

OBJECT                          = TABLE
  INTERCHANGE_FORMAT            = BINARY
  ROWS                          = 1425
  COLUMNS                       = 40
  ROW_BYTES                     = 1032
  ^STRUCTURE                    = "ADFTBL.FMT"
END_OBJECT                      = TABLE
END
```

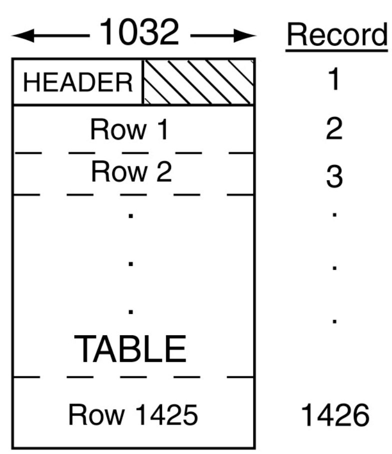
Contents of format file "ADFTBL.FMT":

```
OBJECT                          = COLUMN
  NAME                          = SFDU_LABEL_AND_LENGTH
  START_BYTE                    = 1
  DATA_TYPE                     = CHARACTER
  BYTES                         = 20
  UNIT                          = "N/A"
  DESCRIPTION                   = "The SFDU_label_and_length element
      identifies the label and length of the Standard Format Data Unit
      (SFDU)."
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
  NAME                          = FOOTPRINT_NUMBER
  START_BYTE                    = 21
  DATA_TYPE                     = LSB_INTEGER
  BYTES                         = 4
  UNIT                          = "N/A"
  DESCRIPTION                   = "The footprint_number element provides
    a signed integer value. The altimetry and radiometry processing
    program assigns footprint 0 to that observed at nadir at periapsis.
    The remaining footprints are located along the spacecraft nadir
    track, with a separation that depends on the Doppler resolution of
    the altimeter at the epoch at which that footprint is observed. Pre-
    periapsis footprints will be assigned negative numbers, post-
    periapsis footprints will be assigned positive ones. A loss of
```

```
    several consecutive burst records from the ALT-EDR will result in
    missing footprint numbers."
END_OBJECT                    = COLUMN

...

OBJECT                        = COLUMN
  NAME                        = DERIVED_THRESH_DETECTOR_INDEX
  START_BYTE                  = 1001
  DATA_TYPE                   = LSB_UNSIGNED_INTEGER
  BYTES                       = 4
  UNIT                        = "N/A"
  DESCRIPTION                 = "The derived_thresh_detector_index
    element provides the value of the element in
    range_sharp_echo_profile that satisfies the altimeter threshold
    detection algorithm, representing the distance to the nearest object
    in this radar footprint in units of 33.2 meters, modulus a 10.02
    kilometer altimeter range ambiguity."
END_OBJECT                    = COLUMN
```

A.29.5.1.4 Example: PDS Recommended Method for Dealing with Odd-Sized Headers

The preceding format may be difficult to deal with in some cases because of the odd-sized header preceeding the TABLE object. The recommended approach to this situation is pad the HEADER object out to an integral multiple of the TABLE row size, and then let RECORD_BYTES = ROW_BYTES. Modifying the above case accordingly would yield the following:

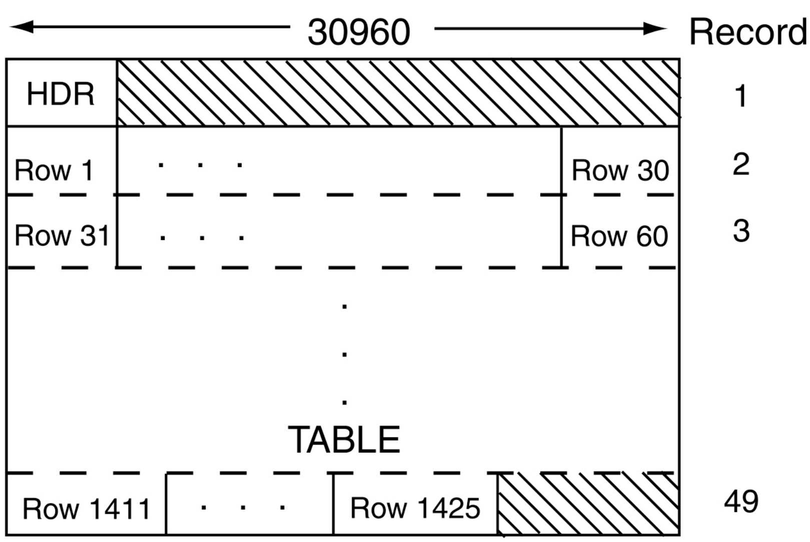



```
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 1032
FILE_RECORDS                  = 1426
^HEADER                       = ("ADF01141.3",1)
^TABLE                        = ("ADF01141.3",2)
 ...

/* DATA OBJECT DEFINITIONS */
```

```
OBJECT                        = HEADER
 HEADER_TYPE                  = SFDU
 BYTES                        = 500
END_OBJECT

OBJECT                        = TABLE
 INTERCHANGE_FORMAT           = BINARY
 ROWS                         = 1425
 COLUMNS                      = 40
 ROW_BYTES                    = 1032
 ^STRUCTURE                   = "ADFTBL.FMT"
END_OBJECT
END
```

A.29.5.1.5 Alternate Format – Rows on Record Boundaries

The following label fragment and illustration provide a second alternate data organization for the preceding example. In this example, a record size of 30,960 is used to hold 30 rows of the TABLE. Again the 500-byte HEADER uses only a portion of the first record.




```
...
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 30960
FILE_RECORDS                  = 49
^HEADER                       = ("ADF01141.3",1)
^TABLE                        = ("ADF01141.3")
...

/* DATA OBJECT DEFINITIONS */

OBJECT                        = HEADER
  HEADER_TYPE                 = SFDU
```

```
  BYTES                         = 500
END_OBJECT                      = HEADER

OBJECT                          = TABLE
  INTERCHANGE_FORMAT            = BINARY
  ROWS                          = 1425
  COLUMNS                       = 40
  ROW_BYTES                     = 1032
  ^STRUCTURE                    = "ADFTBL.FMT"
END_OBJECT                      = TABLE
```

A.29.5.2 Multiple TABLEs in a Single Data File

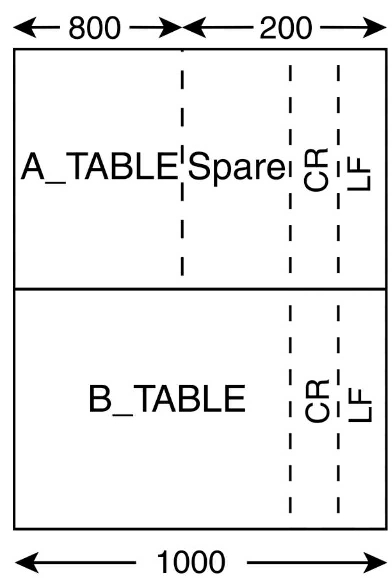
A single data product file may contain several ASCII or binary TABLEs, each with a different logical row size. There are several possible approaches to formatting such a product file, depending on whether the product contains binary or ASCII data. When all the TABLEs in the data file are ASCII tables there are two formatting options: fixed-length file records or stream records. When binary data are involved, even if only in a single TABLE, fixed-length file records are mandatory.

A.29.5.2.1 Example: Multiple ASCII tables – Fixed-Length Records

In the case of a series of ASCII TABLE objects with varying ROW_BYTES values, a fixed-length record file may be generated by padding all rows of all TABLEs out to the length of the longest rows by adding blank characters between the end of the last COLUMN and the <CR><LF> record delimiters.

When this approach is used, RECORD_TYPE is FIXED_LENGTH and RECORD_BYTES = ROW_BYTES.

```
RECORD_TYPE             = FIXED_LENGTH
RECORD_BYTES            = 1000
...
OBJECT                  = A_TABLE
  INTERCHANGE_FORMAT    = ASCII
  ROW_BYTES             = 1000
  ...
END_OBJECT              = A_TABLE

OBJECT                  = B_TABLE
  INTERCHANGE_FORMAT    = ASCII
  ROW_BYTES             = 1000
  ...
END_OBJECT              = B_TABLE
```



Note that each TABLE object has the same value of ROW_BYTES, even though in the smaller table the rightmost bytes will be ignored. Alternately, the filler bytes may be documented as ROW_SUFFIX_BYTES. Say, for example, that in the above case B_TABLE only required 780 bytes for its rows. The following definition for B_TABLE marks the last 220 bytes of each row as suffix bytes:

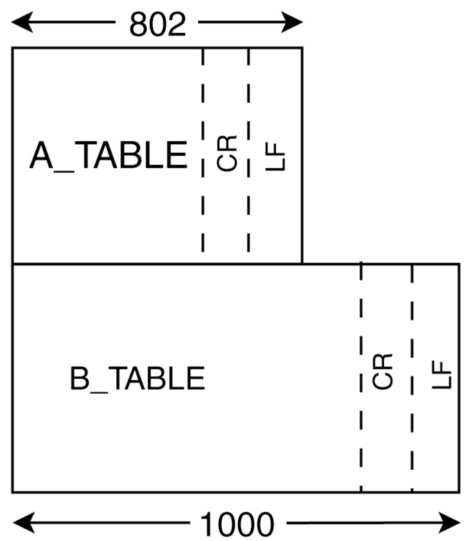
```
OBJECT                        = B_TABLE
  INTERCHANGE_FORMAT          = ASCII
  ROW_BYTES                   = 780
  ROW_SUFFIX_BYTES            = 220
  ...
```

A.29.5.2.2 END_OBJECT = B_TABLE

A.29.5.2.2 Example: Multiple ASCII tables – Stream Records

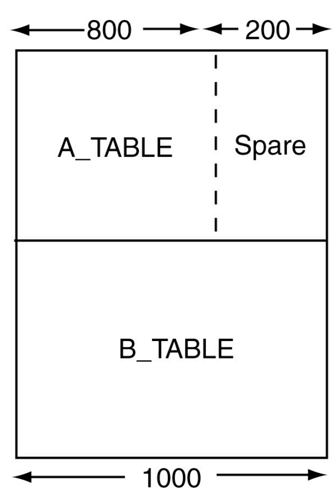
Sometimes padding TABLE records out to a common fixed length creates more problems than it solves. When this is true each TABLE should retain its own ROW_BYTES value, without padding, and the file RECORD_TYPE is set to STREAM. RECORD_BYTES should be omitted. The following label fragment illustrates this situation.

```
RECORD_TYPE                = STREAM
...

OBJECT                     = A_TABLE
  INTERCHANGE_FORMAT       = ASCII
  ROW_BYTES                = 802
  ...
END_OBJECT                 = A_TABLE

OBJECT                     = B_TABLE
  INTERCHANGE_FORMAT       = ASCII
  ROW_BYTES                = 1000
  ...
END_OBJECT                 = B_TABLE
```



A.29.5.2.3 Example: Multiple Binary Tables – Fixed-Length Records

When binary data are involved the file records *must* be fixed-length. The records of the smaller TABLE(s) are padded, usually with null characters, out to the maximum ROW_BYTES value in the file. The padding bytes are accounted for in the TABLE definition using one of two methods: either by defining a COLUMN called “SPARE” to define the number and location of these spare bytes, or by using the ROW_SUFFIX_BYTES keyword, as in the case of multiple ASCII tables. In the following example, the first table, A_TABLE, has a logical row length of 800 bytes. Each row has been padded out to 1000 bytes, the length of the B_TABLE rows, with a 200-byte SPARE column:




```
RECORD_TYPE                = FIXED_LENGTH
RECORD_BYTES               = 1000
```

```
...

OBJECT                    = A_TABLE
  INTERCHANGE_FORMAT      = BINARY
  ROW_BYTES               = 1000
  ...

  OBJECT                  = COLUMN
    NAME                  = "TIME TAG"
    DATA_TYPE             = TIME
    START_BYTE            = 1
    BYTES                 = 23
  END_OBJECT              = COLUMN
  ...
  OBJECT                  = COLUMN
    NAME                  = "SPARE"
    DATA_TYPE             = "N/A"
    START_BYTE            = 801
    BYTES                 = 200
  END_OBJECT              = COLUMN
END_OBJECT                = A_TABLE

OBJECT                    = B_TABLE
  INTERCHANGE_FORMAT      = BINARY
  ROW_BYTES               = 1000
  ...
END_OBJECT                = B_TABLE
```

A.29.5.3 ROW_PREFIX or ROW_SUFFIX Use

ROW_PREFIX_BYTES and ROW_SUFFIX_BYTES are provided for dealing with two situations:

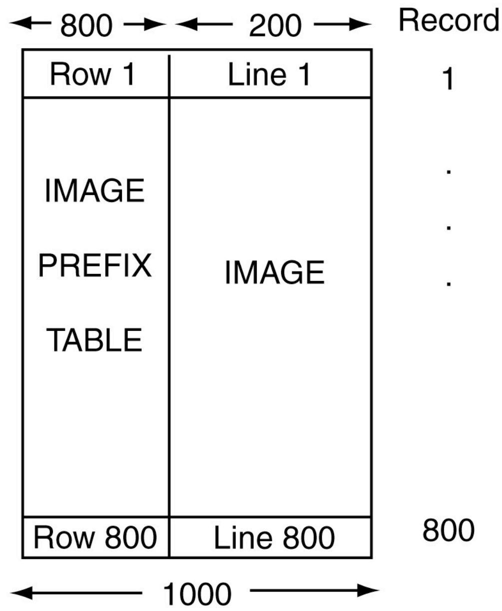
1. When a TABLE object is stored in parallel with another data object, frequently an IMAGE. In this case, each physical record of the file contains a TABLE row as one part of the record and an IMAGE line as the other part.
2. When a TABLE has had each of its rows padded out to a fixed length larger than the logical row size of the table.

Each method is illustrated below.

A.29.5.3.1 Example: Parallel TABLE and IMAGE objects

The following label fragment illustrates a file with fixed-length records, each of which contains one row of a TABLE data object and one line of an IMAGE object. This is a common format for providing ancillary information applicable to each IMAGE line. In the TABLE object the bytes belonging to the IMAGE are accounted for as ROW_SUFFIX_BYTES. In the IMAGE object the bytes belonging to the TABLE row are accounted for as LINE_PREFIX_BYTES.

```
RECORD_TYPE          = FIXED_LENGTH
RECORD_BYTES         = 1000
...

OBJECT               = TABLE
  INTERCHANGE_FORMAT= BINARY
  ROW_BYTES          = 200
  ROW_SUFFIX_BYTES   = 800
  ...
END_OBJECT           = TABLE

OBJECT               = IMAGE
  LINE_SAMPLES       = 800
  SAMPLE_BITS        = 8
  LINE_PREFIX_BYTES  = 200
  ...
END_OBJECT           = IMAGE
```



Note that in each object the total size of the logical record plus all prefix and suffix bytes is equal to the file record size. That is:

RECORD_BYTES = ROW_BYTES + ROW_PREFIX_BYTES + ROW_SUFFIX_BYTES

and

RECORD_BYTES = (LINE_SAMPLES * SAMPLE_BITS / 8) + ROW_PREFIX_BYTES + ROW_SUFFIX_BYTES

A.29.5.4 CONTAINER Object use

Complex TABLEs may contain a set of columns of different data types which repeat a number of times in the row. In this case a CONTAINER object, which groups a set of inhomogeneous COLUMN objects, may be used to provide a single definition for the repeating group. Section A.8 contains an example of a TABLE object which makes use of a CONTAINER object.

A.29.5.5 Guidelines for SPARE fields

Some TABLE objects contain spare bytes embedded in the record but not included in any COLUMN object definition. They may be there for spacing or alignment purposes, or they may have been reserved in the original data record for future use. Regardless of their origin, PDS recommends that all such spare bytes be documented as COLUMNs in the TABLE definition in the interests of documentation and validation. Spare bytes may be included in both binary and ASCII table objects. Guidelines for dealing with spare bytes in both cases follow.

A.29.5.6 SPARE fields - Binary Tables

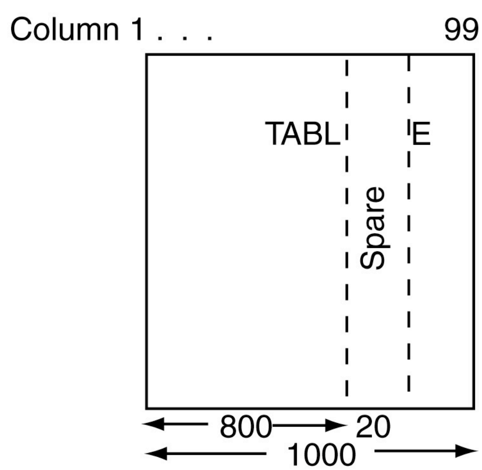
The following guidelines apply to spare byte fields in binary table objects:

- Embedded spare fields must be explicitly defined in COLUMN objects, except when the spare field appears at the beginning or end of a row where ROW_PREFIX_BYTES or ROW_SUFFIX_BYTES is used.
- Spare COLUMNs must have DATA_TYPE = “N/A”.
- Multiple spare COLUMNs may all specify NAME = “SPARE”.
- Spare bytes may occur as prefix or suffix bytes in the rows.
- Prefix or suffix spares may be identified either with a spare COLUMN object or by use of ROW_PREFIX_BYTES or ROW_SUFFIX_BYTES

The following examples illustrate the various situations.

A.29.5.6.1 Example: SPARE field embedded in a Binary TABLE

In the following label fragment, a spare column defines a series of bytes reserved for future use in the middle of the data record:

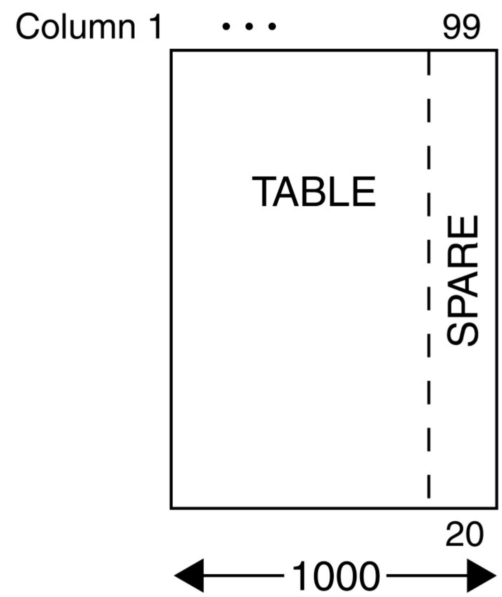



```
RECORD_TYPE           = FIXED_LENGTH
RECORD_BYTES          = 1000
...

OBJECT                = TABLE
  INTERCHANGE_FORMAT  = BINARY
  ROW_BYTES           = 1000
  COLUMNS             = 99
  ...

  OBJECT              = COLUMN
    NAME              = SPARE
    COLUMN_NUMBER     = 87
    START_BYTE        = 793
    BYTES             = 21
    DATA_TYPE         = "N/A"
    DESCRIPTION       = "Reserved for future user by Mission Ops."
  END_OBJECT          = COLUMN
  ...

  OBJECT              = COLUMN
    ...

END_OBJECT            = TABLE
```

A.29.5.6.2 Example: Spares at end of a Binary TABLE – Explicit 'SPARE' Column

In this label fragment, spare bytes have been included on the end of each record of the table. These bytes are described as an additional COLUMN at the end of the record.

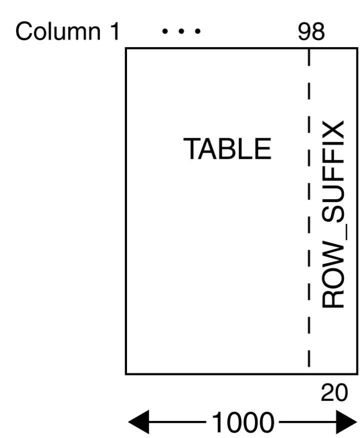
```
RECORD_TYPE           = FIXED_LENGTH
RECORD_BYTES          = 1000
...

OBJECT                = TABLE
  INTERCHANGE_FORMAT  = BINARY
  ROW_BYTES           = 1000
  COLUMNS             = 99
  ...

  OBJECT              = COLUMN
    COLUMN_NUMBER     = 1
    NAME              = "TIME TAG"
    ...
  END_OBJECT
  ...

  OBJECT              = COLUMN
    COLUMN_NUMBER     = 99
    NAME              = SPARE
    BYTES             = 20
    DATA_TYPE         = "N/A"
    START_BYTE        = 981
    ...
  END_OBJECT          = COLUMN
END_OBJECT            = TABLE
```



A.29.5.6.3 Example: Spares at end of a Binary TABLE - ROW_SUFFIX_BYTES use

This fragment illustrates the same physical file layout as the previous example, but in this case the spare bytes are defined using the ROW_SUFFIX_BYTES keyword, rather than defining an additional spare COLUMN.

```
RECORD_TYPE           = FIXED_LENGTH
RECORD_BYTES          = 1000
...

OBJECT                = TABLE
  INTERCHANGE_FORMAT  = BINARY
  ROW_BYTES           = 980
  ROW_SUFFIX_BYTES    = 20
  COLUMNS             = 98
  ...
END_OBJECT            = TABLE
```



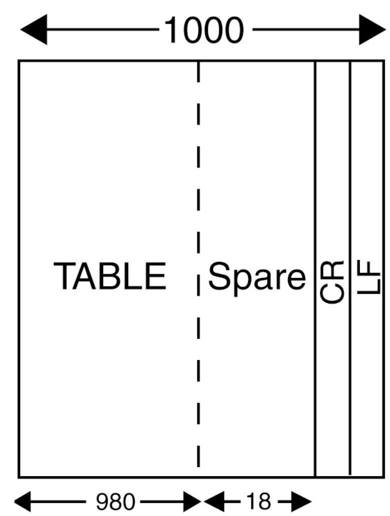
A.29.5.7 SPARE fields - ASCII Tables with Fixed Length Records

In ASCII tables, field delimiters (") and (,) and the <CR><LF> pair are considered part of the data, even though the COLUMN objects attributes do not include them. Spare bytes in ASCII tables may contain only the blank character (ASCII decimal code 32). The following guidelines apply to spare byte fields in ASCII table objects:

- Embedded spares are not allowed.
- Spares are allowed at the end of each row of data.
- The <CR><LF> follows the spare data.
- There are no delimiters (commas or quotes) surrounding the spares.
- Spares at the end of the data can be ignored (like field delimiters and <CR><LF>) or they can be identified
  (1) in the Table DESCRIPTION; or
  (2) by using ROW_SUFFIX_BYTES (note that these bytes should not be included in the value of ROW_BYTES)

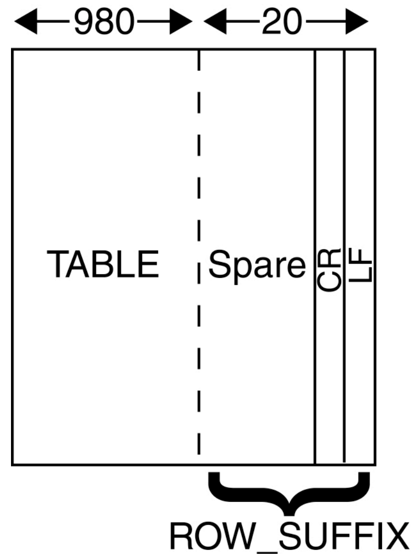
A.29.5.7.1 Example - SPARE field at end of ASCII TABLE - Table description note

```
RECORD_TYPE             = FIXED_LENGTH
RECORD_BYTES            = 1000
...

OBJECT                  = TABLE
  INTERCHANGE_FORMAT    = ASCII
  ROW_BYTES             = 1000
  ...

DECRIPTION              ="This table contains 980
      bytes of table data followed by 18 bytes of
      blank spares. Bytes 999 and 1000 contain the
      <CR><LF> pair."
```



A.29.5.7.2 Example - Spares at end of a ASCII TABLE - ROW_SUFFIX use

```
RECORD_TYPE              = FIXED_LENGTH
RECORD_BYTES             = 1000
...

OBJECT                   = TABLE
  INTERCHANGE_FORMAT     = ASCII
  ROW_BYTES              = 980
  ROW_SUFFIX_BYTES       = 20
  ...
DECRIPTION              ="This table contains
      980 bytes of table data followed by 20
      bytes of spare data of which the last
      two bytes, bytes 999 and 1000, contain
      the <CR><LF> pair."
END_OBJECT               = TABLE
```



A.29.5.8 SPARE fields - ASCII Tables with STREAM Records

Spare fields are not used with ASCII Tables in STREAM record formats. In STREAM files, the last data field explicitly defined with a COLUMN object is followed immediately by the <CR><LF> pair. Since there is no use for spares at the end of the data, and embedded spares are not allowed in ASCII tables, spares are not applicable here.

A.30 TEXT

The TEXT object describes a file which contains plain text. It is most often used in an attached label, so that the text begins immediately after the END statement of the label. PDS recommends that TEXT objects contain no special formatting characters, with the exception of the carriage return/line feed sequence and the page break. Tab characters are discouraged, since they are interpreted differently by different programs.

Use of the carriage-return/line-feed sequence (<CR><LF>) is required for cross-platform support. PDS further recommends that text lines be limited to 80 characters, with delimiters, to facilitate visual inspection and printing of text files.

NOTE: The TEXT object is most often used for files describing the contents of an archive volume or the contents of a directory, such as AAREADME.TXT, DOCINFO.TXT, VOLINFO.TXT, SOFTINFO.TXT, etc. These files must be in plain, unmarked ASCII text and always have a file extension of ".TXT". Documentation files that are in plain ASCII text, on the other hand, must be described using the DOCUMENT object. (See the definition of the DOCUMENT Object in Section A.12.)

The required NOTE field should provide a brief introduction to the TEXT.

A.30.1 Required Keywords

1. NOTE
2. PUBLICATION_DATE

A.30.2 Optional Keywords

1. INTERCHANGE_FORMAT

A.30.3 Required Objects

None

A.30.4 Optional Objects

None

A.30.5 Example

The example below is a portion of an AAREADME.TXT file.

```
PDS_VERSION_ID                = PDS3
RECORD_TYPE                   = STREAM

OBJECT                        = TEXT
  PUBLICATION_DATE            = 1991-05-28
  NOTE                        = "Introduction to this CD-ROM volume."
END_OBJECT                    = TEXT
END
               GEOLOGIC REMOTE SENSING FIELD EXPERIMENT

This set of compact read-only optical disks (CD-ROMs) contains a data
collection acquired by ground-based and airborne instruments during the
Geologic Remote Sensing Field Experiment (GRSFE).  Extensive
documentation is also included.  GRSFE took place in July, September,
and October, 1989, in the southern Mojave Desert, Death Valley, and the
Lunar Crater Volcanic Field, Nevada.  The purpose of these CD-ROMs is
to make available in a compact form through the Planetary Data System
(PDS) a collection of relevant data to conduct analyses in preparation
for the Earth Observing System (EOS), Mars Observer (MO), and other
missions.  The generation of this set of CD-ROMs was sponsored by the
NASA Planetary Geology and Geophysics Program, the Planetary Data
System (PDS) and the Pilot Land Data System (PLDS).

This AAREADME.TXT file is one of the two nondirectory files located in
the top level directory of each CD-ROM volume in this collection. The
other file, VOLDESC.CAT, contains an overview of the data sets on these
CD-ROMs and is written in a format that is designed for access by
computers.  These two files appear on every volume in the collection.
All other files on the CD-ROMs are located in directories below the top
level directory ....
```

A.31 VOLUME

The VOLUME object describes a physical or logical unit used to store or distribute data products (e.g., a magnetic tape, CD-ROM disk, or floppy disk) that contain directories and files. The directories and files may include documentation, software, calibration and geometry information as well as the actual science data.

A.31.1 Required Keywords

1. DATA_SET_ID
2. DESCRIPTION
3. MEDIUM_TYPE
4. PUBLICATION_DATE
5. VOLUME_FORMAT
6. VOLUME_ID
7. VOLUME_NAME
8. VOLUME_SERIES_NAME
9. VOLUME_SET_NAME
10. VOLUME_SET_ID
11. VOLUME_VERSION_ID
12. VOLUMES

A.31.2 Optional Keywords

1. BLOCK_BYTES
2. DATA_SET_COLLECTION_ID
3. FILES
4. HARDWARE_MODEL_ID
5. LOGICAL_VOLUMES
6. LOGICAL_VOLUME_PATH_NAME
7. MEDIUM_FORMAT
8. NOTE
9. OPERATING_SYSTEM_ID
10. PRODUCT_TYPE
11. TRANSFER_COMMAND_TEXT
12. VOLUME_INSERT_TEXT

A.31.3 Required Objects

1. CATALOG
2. DATA_PRODUCER

A.31.4 Optional Objects

1. DIRECTORY
2. FILE
3. DATA_SUPPLIER

A.31.5 Example 1 (Typical CD-ROM Volume)

Please see the example in Section A.5 for the CATALOG object.

A.31.6 Example 2 (Tape Volume)

The following VOLUME object example shows how directories and files are detailed when a volume is stored on an ANSI tape for transfer. This form of the VOLUME object should be used when transferring volumes of data on media which do not support hierarchical directory structures (for example, when submitting a volume of data on tape for premastering to CDROM). The VOLDESC.CAT file will contain the standard volume keywords, but the values of MEDIUM_TYPE, MEDIUM_FORMAT and VOLUME_FORMAT should indicate that the volume is stored on tape.

In this example two files are defined in the root directory of the volume, VOLDESC.CAT and AAREADME.TXT. The first DIRECTORY object defines the CATALOG directory which contains meta data in the included, individual catalog objects. In this example, all the catalog objects are concatenated into a single file, CATALOG.CAT. The second DIRECTORY object defines an INDEX subdirectory containing three files: INDXINFO.TXT, INDEX.LBL, and INDEX.TAB. Following that directory, the first data directory is defined. Note that the SEQUENCE_NUMBER keyword indicates the physical sequence of the files on the tape volume.

```
PDS_VERSION_ID               = PDS3
OBJECT                       = VOLUME
  VOLUME_SERIES_NAME         = "MISSION TO MARS"
  VOLUME_SET_NAME            = "MARS DIGITAL IMAGE MOSAIC AND DIGITAL
                                TERRAIN MODEL"
  VOLUME_SET_ID              = USA_NASA_PDS_VO_2001_TO_VO_2007
  VOLUMES                    = 7
  VOLUME_NAME                = "MDIM/DTM VOLUME 7: GLOBAL COVERAGE"
  VOLUME_ID                  = VO_2007
  VOLUME_VERSION_ID          = "VERSION 1"
  PUBLICATION_DATE           = 1992-04-01
  DATA_SET_ID                = "VO1/VO2-M-VIS-5-DTM-V1.0"
  MEDIUM_TYPE                = "8-MM HELICAL SCAN TAPE"
  MEDIUM_FORMAT              = "2 GB"
  VOLUME_FORMAT              = ANSI
  HARDWARE_MODEL_ID          = "VAX 11/750"
  OPERATING_SYSTEM_ID        = "VMS 4.6"
```

```
DESCRIPTION                  = "This volume contains the Mars Digital
  Terrain Model and Mosaicked Digital Image Model covering the entire
  planet at resolutions of 1/64 and 1/16 degree/pixel.  The volume
  also contains Polar Stereographic projection files of the north and
  south pole areas from 80 to 90 degrees latitude; Mars Shaded Relief
  Airbrush Maps at 1/16 and 1/ 4 degree/pixel; a gazetteer of Mars
  features; and a table of updated viewing geometry files of the
  Viking EDR images that comprise the MDIM."
MISSION_NAME                 = VIKING
SPACECRAFT_NAME              = {VIKING_ORBITER_1,VIKING_ORBITER_2}
SPACECRAFT_ID                = {VO1,VO2}

OBJECT                       = DATA_PRODUCER
  INSTITUTION_NAME           = "U.S.G.S. FLAGSTAFF"
  FACILITY_NAME              = "BRANCH OF ASTROGEOLOGY"
  FULL_NAME                  = "Eric M. Eliason"
  DISCIPLINE_NAME            = "IMAGE PROCESSING"
  ADDRESS_TEXT               = "Branch of Astrogeology
                                United States Geological Survey
                                2255 North Gemini Drive
                                Flagstaff, Arizona. 86001 USA"
END_OBJECT                   = DATA_PRODUCER

OBJECT                       = CATALOG
  ^CATALOG                   = "CATALOG.CAT"
END_OBJECT                   = CATALOG

OBJECT                       = FILE
  FILE_NAME                  = "VOLDESC.CAT"
  RECORD_TYPE                = STREAM
  SEQUENCE_NUMBER            = 1
END_OBJECT                   = FILE

OBJECT                       = FILE
  FILE_NAME                  = "AAREADME.TXT"
  RECORD_TYPE                = STREAM
  SEQUENCE_NUMBER            = 2
END_OBJECT                   = FILE

OBJECT                       = DIRECTORY
  NAME                       = CATALOG

  OBJECT                     = FILE
    FILE_NAME                = "CATALOG.CAT"
    RECORD_TYPE              = STREAM
    SEQUENCE_NUMBER          = 3
  END_OBJECT                 = FILE
END_OBJECT                   = DIRECTORY

OBJECT                       = DIRECTORY
  NAME                       = DOCUMENT


  OBJECT                     = FILE
```

```
      FILE_NAME               = "VOLINFO.TXT"
      RECORD_TYPE             = STREAM
      SEQUENCE_NUMBER         = 4
      END_OBJECT              = FILE

    OBJECT                    = FILE
      FILE_NAME               = "DOCINFO.TXT"
      RECORD_TYPE             = STREAM
      SEQUENCE_NUMBER         = 5
    END_OBJECT                = FILE
  END_OBJECT                  = DIRECTORY

  OBJECT                      = DIRECTORY
    NAME                      = INDEX

    OBJECT                    = FILE
      FILE_NAME               = "INDXINFO.TXT"
      RECORD_TYPE             = STREAM
      SEQUENCE_NUMBER         = 6
    END_OBJECT                = FILE

    OBJECT                    = FILE
      FILE_NAME               = "INDEX.LBL"
      RECORD_TYPE             = STREAM
      SEQUENCE_NUMBER         = 7
    END_OBJECT                = FILE

    OBJECT                    = FILE
      FILE_NAME               = "INDEX.TAB"
      RECORD_TYPE             = FIXED_LENGTH
      RECORD_BYTES            = 512
      FILE_RECORDS            = 6822
      SEQUENCE_NUMBER         = 8
    END_OBJECT                = FILE
  END_OBJECT                  = DIRECTORY

  OBJECT                      = DIRECTORY
    NAME                      = MG00NXXX

    OBJECT                    = FILE
      FILE_NAME               = "MG00N012.IMG"
      RECORD_TYPE             = FIXED_LENGTH
      RECORD_BYTES            = 964
      FILE_RECORDS            = 965
      SEQUENCE_NUMBER         = 10
    END_OBJECT                = FILE
  END_OBJECT                  = DIRECTORY

END_OBJECT                    = VOLUME
END
```

A.31.7 **Example 3** (Logical Volumes in an Archive Volume)

The following examples illustrate the use of the VOLUME object in the top level and at the logical volume level of an archive volume. Note that the VOLUME object is *required* at both levels.

In these examples, the CD-ROM is structured as three separate logical volumes with root directories named PPS/, UVS/ and RSS/. An additional SOFTWARE directory is supplied at volume root for use with all logical volumes.

A.31.7.1 Logical Volumes – Volume Object (root level)

The example below, illustrates the use of the VOLUME object at the top level of a CD-ROM (i.e., a physical volume) containing several logical volumes. Note the values of the keywords DATA_SET_ID, LOGICAL_VOLUMES, and LOGICAL_VOLUME_PATH_NAME, which list the complete set of values relevant to this volume.

```
PDS_VERSION_ID                = PDS3
OBJECT                        = VOLUME
  VOLUME_SERIES_NAME          = "VOYAGERS TO THE OUTER PLANETS"
  VOLUME_SET_NAME             = "PLANETARY RING OCCULTATIONS FROM
                                 VOYAGER"
  VOLUME_SET_ID               = "USA_NASA_PDS_VG_3001"
  VOLUMES                     = 1
  MEDIUM_TYPE                 = "CD-ROM"
  VOLUME_FORMAT               = "ISO-9660"
  VOLUME_NAME                 = "VOYAGER PPS/UVS/RSS RING OCCULTATIONS"
  VOLUME_ID                   = "VG_3001"
  VOLUME_VERSION_ID           = "VERSION 1"
  PUBLICATION_DATE            = 1994-03-01
  DATA_SET_ID                 = {"VG2-SR/UR/NR-PPS-4-OCC-V1.0",
                                 "VG1/VG2-SR/UR/NR-UVS-4-OCC-V1.0",
                                 "VG1/VG2-SR/UR/NR-RSS-4-OCC-V1.0"}
  LOGICAL_VOLUMES             = 3
  LOGICAL_VOLUME_PATH_NAME    = {"PPS/", "UVS/", "RSS/"}
  DESCRIPTION                 = "This volume contains the Voyager 1 and
    Voyager 2 PPS/UVS/RSS ring occultation and ODR data sets. Included
    are data files at a variety of levels of processing, plus ancillary
    geometry, calibration and trajectory files plus software and
    documentation.

    This CD-ROM is structured as three separate logical volumes with
    root directories named PPS/, UVS/ and RSS/. An additional SOFTWARE
    directory is supplied at volume root for use with all logical
    volumes."

  OBJECT                      = DATA_PRODUCER
    INSTITUTION_NAME          = "PDS RINGS NODE"
    FACILITY_NAME             = "NASA AMES RESEARCH CENTER"
    FULL_NAME                    = "DR. MARK R. SHOWALTER"
    DISCIPLINE_NAME              = "RINGS"
    ADDRESS_TEXT                 = "Mail Stop 245-3
```

```
                                   NASA Ames Research Center
                                   Moffett Field, CA 94035-1000"
    END_OBJECT                    = DATA_PRODUCER

    OBJECT                        = CATALOG
      DATA_SET_ID                 = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
      LOGICAL_VOLUME_PATH_NAME    = "PPS/"
      ^MISSION_CATALOG            = "MISSION.CAT"
      ^INSTRUMENT_HOST_CATALOG    = "INSTHOST.CAT"
      ^INSTRUMENT_CATALOG         = "INST.CAT"
      ^DATA_SET_COLLECTION_CATALOG = "DSCOLL.CAT"
      ^DATA_SET_CATALOG           = "DATASET.CAT"
      ^REFERENCE_CATALOG          = "REF.CAT"
      ^PERSONNEL_CATALOG          = "PERSON.CAT"
    END_OBJECT                    = CATALOG

    OBJECT                        = CATALOG
      DATA_SET_ID                 = "VG1/VG2-SR/UR/NR-UVS-4-OCC-V1.0"
      LOGICAL_VOLUME_PATH_NAME    = "UVS/"
      ^MISSION_CATALOG            = "MISSION.CAT"
      ^INSTRUMENT_HOST_CATALOG    = "INSTHOST.CAT"
      ^INSTRUMENT_CATALOG         = "INST.CAT"
      ^DATA_SET_COLLECTION_CATALOG = "DSCOLL.CAT"
      ^DATA_SET_CATALOG           = "DATASET.CAT"
      ^REFERENCE_CATALOG          = "REF.CAT"
      ^PERSONNEL_CATALOG          = "PERSON.CAT"
    END_OBJECT                    = CATALOG

    OBJECT                        = CATALOG
      DATA_SET_ID                 = "VG1/VG2-SR/UR/NR-RSS-4-OCC-V1.0"
      LOGICAL_VOLUME_PATH_NAME    = "RSS/"
      ^MISSION_CATALOG            = "MISSION.CAT"
      ^INSTRUMENT_HOST_CATALOG    = "INSTHOST.CAT"
      ^INSTRUMENT_CATALOG         = "INST.CAT"
      ^DATA_SET_COLLECTION_CATALOG = "DSCOLL.CAT"
      ^DATA_SET_CATALOG           = "DATASET.CAT"
      ^REFERENCE_CATALOG          = "REF.CAT"
      ^PERSONNEL_CATALOG          = "PERSON.CAT"
    END_OBJECT                    = CATALOG

  END_OBJECT                      = VOLUME
  END
```

A.31.7.2 Logical Volumes – Volume Object (logical volume level)

The example below, illustrates the use of the VOLUME object required at the top level of a logical volume. Note that at this level the keywords DATA_SET_ID and LOGICAL_VOLUME_PATH_NAME contain only the values relevant to the current logical volume. Also, the keyword LOGICAL_VOLUMES does not appear here.

```
  PDS_VERSION_ID                  = PDS3
  OBJECT                          = VOLUME
    VOLUME_SERIES_NAME            = "VOYAGERS TO THE OUTER PLANETS"
```

```
  VOLUME_SET_NAME                = "PLANETARY RING OCCULTATIONS
                                   FROM VOYAGER"
  VOLUME_SET_ID                  = "USA_NASA_PDS_VG_3001"
  VOLUMES                        = 1
  MEDIUM_TYPE                    = "CD-ROM"
  VOLUME_FORMAT                  = "ISO-9660"
  VOLUME_NAME                    = "VOYAGER PPS/UVS/RSS RING
                                   OCCULTATIONS"
  VOLUME_ID                      = "VG_3001"
  VOLUME_VERSION_ID              = "VERSION 1"
  PUBLICATION_DATE               = 1994-03-01
  DATA_SET_ID                    = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
  LOGICAL_VOLUME_PATH_NAME       = "PPS/"
  DESCRIPTION                    = "This logical volume contains the
   Voyager 2 PPS ring occultation data sets. Included are data files at
   a variety of levels of processing, plus ancillary geometry,
   calibration  and trajectory files plus software and documentation."

  OBJECT                         = DATA_PRODUCER
    INSTITUTION_NAME             = "PDS RINGS NODE"
    FACILITY_NAME                = "NASA AMES RESEARCH CENTER"
    FULL_NAME                    = "DR. MARK R. SHOWALTER"
    DISCIPLINE_NAME              = "RINGS"
    ADDRESS_TEXT                 = "Mail Stop 245-3
                                   NASA Ames Research Center
                                   Moffett Field, CA 94035-1000"
  END_OBJECT                     = DATA_PRODUCER

  OBJECT                         = CATALOG
    DATA_SET_ID                  = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
    LOGICAL_VOLUME_PATH_NAME     = "PPS/"
    ^MISSION_CATALOG             = "MISSION.CAT"
    ^INSTRUMENT_HOST_CATALOG     = "INSTHOST.CAT"
    ^INSTRUMENT_CATALOG          = "INST.CAT"
    ^DATA_SET_COLLECTION_CATALOG = "DSCOLL.CAT"
    ^DATA_SET_CATALOG            = "DATASET.CAT"
    ^REFERENCE_CATALOG           = "REF.CAT"
    ^PERSONNEL_CATALOG           = "PERSON.CAT"
  END_OBJECT                     = CATALOG

END_OBJECT                       = VOLUME
END
```

A.32 WINDOW

The WINDOW object identifies a rectangular area of interest within an IMAGE object. WINDOW objects may not serve as the primary object in a data product, nor may they appear outside the context of an IMAGE object. The areas described by separate WINDOW objects may overlap in whole or in part, but WINDOW object definitions may not be nested.

The boundaries and physical attributes of the WINDOW object are always determined with reference to the enclosing (parent) IMAGE object. That is, "first" is defined with respect to the LINE_DISPLAY_DIRECTION and SAMPLE_DISPLAY_DIRECTION of the IMAGE and the WINDOW must have the same SAMPLE_TYPE and SAMPLE_BITS as the IMAGE. WINDOW objects may not have prefix or suffix bytes.

A.32.1 Required Keywords

1. FIRST_LINE
2. FIRST_LINE_SAMPLE
3. LINES
4. LINE_SAMPLES
5. DESCRIPTION

A.32.2 Optional Keywords

1. NAME
2. TARGET_NAME
3. psdd

A.32.3 Required Objects

None

A.32.4 Optional Objects

None

A.32.5 Example

This example is extracted from an early draft of a Stardust Mission data file. In this case the WINDOW object is being used to identify the section of a sparse IMAGE object which contains actual data.

```
OBJECT                          = IMAGE
  LINES                         = 1024
  LINE_SAMPLES                  = 1024
  SAMPLE_TYPE                   = MSB_UNSIGNED_INTEGER
  SAMPLE_BITS                   = 16
  SAMPLE_BIT_MASK               = 2#0000111111111111#
  MAXIMUM                       = 2877
  MINIMUM                       = 0
  LINE_PREFIX_BYTES             = 20
  LINE_SUFFIX_BYTES             = 24
  PUBLICATION_DATE              = 1994-03-01
  MEAN                          = 63.7351
  STANDARD_DEVIATION            = 174.729
  SATURATED_PIXELS              = 0
  CHECKSUM                      = 66831091

  OBJECT                        = WINDOW
    DESCRIPTION                 = "Stellar image"
    TARGET_NAME                 = "VEGA"
    FIRST_LINE                  = 336
    FIRST_LINE_SAMPLE           = 336
    LINES                       = 351
    LINE_SAMPLES                = 351
  END_OBJECT                    = WINDOW
END_OBJECT                      = IMAGE
```

(This page intentionally left blank.)

Appendix B. Complete PDS Catalog Object Set

This appendix provides a complete set of the PDS catalog objects. Each section includes a description of the object, lists of keywords and sub-objects, guidelines to follow in assigning values, and a specific example of the object. The catalog objects provide high-level information suitable for loading a database to facilitate searches across data sets, collections and volumes.

The catalog objects included on a PDS volume also provide local, high-level documentation. The full set of catalog objects is required in the CATALOG directory of every PDS archive volume. See the *File Specification and Naming* chapter of this document for pointer and file names used with catalog objects.

Not every object described in this section is required in all cases. A PDS Central Node Data Engineer will supply a set of blank catalog object templates to be completed for any specific delivery, and can also supply additional examples if desired.

Description Field Formatting

The examples in the following sections conform to the current recommendations with respect to format and content. Lines in descriptive text fields (DATA_SET_DESC, INSTRUMENT_DESC, etc.) should not exceed 80 characters, including the <CR><LF> line delimiters. The underlining convention for headings and subheadings provide organization levels for human readers and auto-formatting routines:

Heading	Heading Indent	Text Indent	Underscoring Character
Primary	2 characters	4 characters	=
Secondary	4 characters	6 characters	–

Primary, or main, headings are double-underlined through the use of the equal-sign key (=) which corresponds to ASCII decimal 61. Secondary, or subheadings, are single-underlined through the use of the hyphen key (-) which corresponds to ASCII decimal 45. This underlining convention enhances legibility, and in the future will facilitate the creation of hypertext links.

Also, PDS has adopted a convention for indenting primary headings, secondary headings, and textual descriptions to facilitate readability and to make a better presentation. Primary headings start at Column 3. Text under primary headings and secondary headings starts at Column 5. Text under secondary headings starts at Column 7.

Again for ease of readability, there should be 2 blank lines before the start of a primary or secondary heading. If a secondary heading immediately follows a primary heading, then only 1 blank line should separate the secondary heading from the primary heading.

PDS has developed a Windows based program (FORMAT70) that will automatically format the description fields of any catalog template.

Following is a template layout for a DATA_SET_DESC field. This example assumes the keyword DATA_SET_DESC itself starts in the first byte.

```
         1         2         3         4         5         6         7
123456789012345678901234567890123456789012345678901234567890123456789012

DATA_SET_DESC                   = "
(blank line)
(blank line)
  Primary Heading - starts at Column 3
  ====================================
    Text under headings start at Column 5
    more text starting at Column 5...
(blank line)
(blank line)
     Secondary Heading - starts at Column 5
    --------------------------------------
      Text under subheadings start at Column 7
      more text starting at Column 7...
(blank line)
(blank line)
  Primary Heading - starts at Column 3
  ====================================
(blank line)
    Secondary Heading - starts at Column 5
    --------------------------------------
      Text under subheadings start at Column 7
      more text starting at Column 7...
```

Order of Keywords and Sub-Objects

The examples in the following sections illustrate the preferred ordering for keywords and sub-objects. The order used provides a logical flow that makes the catalog files somewhat easier for a human reader to follow.

Chapter Contents

Appendix B. Complete PDS Catalog Object Set B-1

B.1 DATA_SET B-4
B.2 DATA_SET_COLL_ASSOC_DATA_SETS B-11
B.3 DATA_SET_COLLECTION_REF_INFO B-12
B.4 DATA_SET_COLLECTION B-13
B.5 DATA_SET_COLLECTION_INFO B-16
B.6 DATA_SET_HOST B-18
B.7 DATA_SET_INFORMATION B-19
B.8 DATA_SET_MAP_PROJECTION B-22
B.9 DATA_SET_MAP_PROJECTION_INFO B-25
B.10 DATA_SET_MISSION B-27
B.11 DATA_SET_REFERENCE_INFORMATION B-28
B.12 DATA_SET_TARGET B-29
B.13 DS_MAP_PROJECTION_REF_INFO B-30
B.14 IMAGE_MAP_PROJECTION B-31
B.15 INSTRUMENT B-36
B.16 INSTRUMENT_HOST B-41
B.17 INSTRUMENT_HOST_INFORMATION B-43
B.18 INSTRUMENT_HOST_REFERENCE_INFO B-44
B.19 INSTRUMENT_INFORMATION B-45
B.20 INSTRUMENT_REFERENCE_INFO B-48
B.21 INVENTORY B-49
B.22 INVENTORY_DATA_SET_INFO B-51
B.23 INVENTORY_NODE_MEDIA_INFO B-52
B.24 MISSION B-53
B.25 MISSION_HOST B-59
B.26 MISSION_INFORMATION B-60
B.27 MISSION_REFERENCE_INFORMATION B-62
B.28 MISSION_TARGET B-63
B.29 PERSONNEL B-64
B.30 PERSONNEL_ELECTRONIC_MAIL B-66
B.31 PERSONNEL_INFORMATION B-67
B.32 REFERENCE B-68
B.33 SOFTWARE B-75
B.34 SOFTWARE_INFORMATION B-77
B.35 SOFTWARE_ONLINE B-78
B.36 SOFTWARE_PURPOSE B-79
B.37 TARGET B-80
B.38 TARGET_INFORMATION B-82
B.39 TARGET_REFERENCE_INFORMATION B-83

B.1 DATA_SET

The DATA_SET catalog object is used to submit information about a data set to the PDS. The DATA_SET object includes a free-form text description of the data set as well as sub-objects for identifying associated targets, hosts, and references.

B.1.1 Required Keywords

1. DATA_SET_ID

B.1.2 Optional Keywords

None

B.1.3 Required Objects

1. DATA_SET_HOST
2. DATA_SET_INFORMATION
3. DATA_SET_REFERENCE_INFORMATION
4. DATA_SET_TARGET
5. DATA_SET_MISSION

B.1.4 Optional Objects

None

B.1.5 Usage Notes

One DATA_SET_INFORMATION catalog object must be completed for each data set. One DATA_SET_TARGET catalog object must be completed for each target associated with the data set. That is, if there is more than one target, this object is repeated. Similarly, one DATA_SET_HOST catalog object must be completed for each host/instrument pair associated with the data set, and one DATA_SET_REFERENCE_INFORMATION catalog object is required for each individual reference associated with the data set. All references should be included that are relevant to providing more detailed / specific data set information; such as, description of the data set, calibration procedures, processing software, data set documentation, review results, etc. These references may include published articles, books, papers, electronic publications, etc.

Note that the DATA_SET_TARGET, DATA_SET_HOST and DATA_SET_REFERENCE objects associate a particular target, host or reference ID with the data set, but do not themselves define the attributes of the corresponding target, host or reference. For each new ID referenced in one of these fields, a high-level description must be provided in the corresponding catalog object. For example, if the REFERENCE_KEY_ID listed in a DATA_SET_REFERENCE object does not already exist, a new REFERENCE object, defining that REFERENCE_KEY_ID, must also be submitted with the delivery. The Central Node data engineers can assist in locating existing catalog objects that may be referenced in any of the above fields.

B.1.6 Example

```
/* Template: Data Set Template                                    Rev: 1993-09-24  */
/*                                                                                 */
/* Note: Complete one for each data set. Identify multiple targets associated with */
/* the data set by repeating the 3 lines for the DATA_SET_TARGET object.           */
/* Identify multiple hosts associated with the data set by repeating the 4 lines   */
/* for the DATA_SET_HOST object. Identify multiple references associated           */
/* with the data set by repeating the 3 lines of the                               */
/* DATA_SET_REFERENCE_INFORMATION object.                                          */

/* Hierarchy:   DATA_SET                                                           */
/*              DATA_SET_INFORMATION                                               */
/*              DATA_SET_TARGET                                                    */
/*              DATA_SET_HOST                                                      */
/*              DATA_SET_REFERENCE_INFORMATION                                     */

PDS_VERSION_ID                      = PDS3
LABEL_REVISION_NOTE                 = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                         = STREAM

OBJECT                              = DATA_SET
  DATA_SET_ID                       = "MGN-V-RDRS-5-GVDR-V1.0"

  OBJECT                            = DATA_SET_INFORMATION
    DATA_SET_NAME                   = "MGN V RDRS DERIVED GLOBAL VECTOR DATA RECORD V1.0"
    DATA_SET_COLLECTION_MEMBER_FLG  = "N"
    DATA_OBJECT_TYPE                = TABLE
    START_TIME                      = 1990-08-01T00:00:00
    STOP_TIME                       = 1993-12-31T23:59:59
    DATA_SET_RELEASE_DATE           = 1994-07-01
    PRODUCER_FULL_NAME              = "MICHAEL J. MAURER"
    DETAILED_CATALOG_FLAG           = "N"
    DATA_SET_TERSE_DESC             = "The Global Vector Data Record (GVDR)
                                       is a sorted collection of scattering and
                                       emission measurements from the Magellan
                                       Mission"
    ABSTRACT_DESC                   = "The Global Vector Data Record (GVDR) is a sorted
                                      collection of scattering and emission measurements from
                                      the Magellan Mission.  The sorting is into a grid of
                                      equal area 'pixels' distributed regularly about the
                                      planet.  For data acquired from the same pixel but in
                                      different observing geometries, there is a second level
                                      of sorting to accommodate the different geometrical
                                      conditions.  The 'pixel' dimension is 18.225 km.  The
                                      GVDR is presented in Sinusoidal Equal Area (equatorial),
                                      Mercator (equatorial), and Polar Stereographic (polar)
                                      projections.

                                      The GVDR is intended to be the most systematic and
                                      comprehensive representation of the electromagnetic
                                      properties of the Venus surface that can be derived from
                                      Magellan data at this resolution.  It should be useful
                                      in characterizing and comparing distinguishable surface
                                      units."
    CITATION_DESC                   = "Maurer, M.J., MGN V RDRS DERIVED GLOBAL VECTOR
```

```
                                       DATA RECORD V1.0, MGN-V-RDRS-5-GVDR-V1.0, NASA
                                       Planetary Data System, 1994."
  DATA_SET_DESC                    = "
```

Data Set Overview
=================

The Global Vector Data Record (GVDR) is a sorted collection of scattering and emission measurements from the Magellan Mission.The sorting is into a grid of equal area 'pixels' distributed regularly about the planet. For data acquired from the same pixel but in different observing geometries, there is a second level of sorting to accommodate the different geometrical conditions. The 'pixel' dimension is 18.225 km. The GVDR is presented in Sinusoidal Equal Area (equatorial), Mercator (equatorial), and Polar Stereographic (polar) projections.

The GVDR is intended to be the most systematic and comprehensive representation of the electromagnetic properties of the Venus surface that can be derived from Magellan data at this resolution. It should be useful in characterizing and comparing distinguishable surface units.

Parameters
==========

The Magellan data set comprises three basic data types: echoes from the nadir-viewing altimeter (ALT), echoes from the oblique backscatter synthetic aperture radar (SAR) imaging system, and passive radio thermal emission measurements made using the SAR equipment. The objective in compiling the GVDR is to obtain an accurate estimate of the surface backscattering function (sometimes called the specific backscatter function or 'sigma-zero') for Venus from these three data types and to show its variation with incidence (polar) angle, azimuthal angle, and surface location.

The ALT data set has been analyzed to yield profiles of surface elevation [FORD&PETTENGILL1992] and estimates of surface Fresnel reflectivity and estimates of meter-scale rms surface tilts by at least two independent methods [FORD&PETTENGILL1992;TYLER1992]. The 'inversion' approach of [TYLER1992] provides, in addition, an empirical estimate of the surface backscatter function at incidence angles from nadir to as much as 10 degrees from nadir in steps of 0.5 degrees.

Statistical analysis of SAR image pixels for surface regions about 20 km (across track) by 2 km (along track) provided estimates of the surface backscatter function over narrow angular ranges (1-4 degrees) between 15 and 50 degrees from normal incidence [TYLER1992]. By combining results from several orbital passes over the same region in different observing geometries, the backscatter response over the full oblique angular range (15-50) could be compiled. In fact, the number of independent observing geometries attempted with Magellan was limited, and some of these represented changes in azimuth rather than changes in incidence (or polar) angle. Nevertheless, data from many regions were collected in more than one SAR observing geometry. Histograms of pixel values and quadratic fits to the surface backscattering function over narrow ranges of incidence angle were computed by [TYLER1992].

Passive microwave emission by the surface of Venus was measured by the Magellan radar receiver between ALT and SAR bursts. These measurements have been converted to estimates of surface emissivity [PETTENGILLETAL1992]. With certain assumptions the emissivity derived from these data should be the complement of the Fresnel reflectivity derived from the ALT echo strengths.In cases where the two quantities do not add to unity, the assumptions about a simple dielectric (Fresnel) interface at the surface of Venus must be adjusted.

Processing
==========

The processing carried out at the Massachusetts Institute of Technology (MIT) to obtain altimetry profiles and estimates of Fresnel reflectivity and rms surface tilts has been described elsewhere [FORD&PETTENGILL1992]. In brief it involves fitting pre-computed templates to measured echo profiles; the topographic profiles, Fresnel reflectivities, and rms surface tilts are chosen to minimize differences between the data and templates in a least-squares sense. The estimates of emissivity require calibration of the raw data values and correction for attenuation and emission by the Venus atmosphere [PETTENGILLETAL1992]. These data have been collected by orbit number on a set of compact discs [FORD1992] and into a set of global maps, also distributed on compact disc [FORD1993].

At Stanford ALT-EDR tapes were the input for calculation of near-nadir empirical backscattering functions. For oblique backscatter, C-BIDR tapes from the Magellan Project and F-BIDR files obtained via Internet from Washington University were the input products. Output was collected on an orbit-by-orbit basis into a product known as the Surface Characteristics

Vector Data Record (SCVDR). The SCVDR has been delivered to the Magellan Project for orbits through 2599; processing of data beginning with orbit 2600 and continuing through the end-of-Mission is spending completion of the first version of the GVDR.

Data
====

The GVDR data set comprises several 'tables' of results based on analysis of each of the data types described above. These include:

(1) Image Data Table
(2) Radiometry Data Table
(3) MIT ALT Data Table
(4) Stanford ALT Data Table

(1) Image Data Table
This table contains results from analysis of SAR image strips.The results are parameterized by the azimuth angle, the incidence (polar) angle, and the polarization angle. Quantities include the number of image frame lets used to compute the scattering parameters; the median, the mode, and the one-standard-deviation limits of the pixel histogram; and the three coefficients and the reference angle of the quadratic approximation to sigma-zero as a function of incidence angle.

(2) Radiometry Data Table
This table contains results from MIT analysis of the radiometry data. The results are parameterized by the azimuth angle, the incidence angle, and the polarization angle. The results include the number of radiometry footprints used to compute the estimate of thermal emissivity, the emissivity, and its variance.

(3) MIT ALT Data Table
This table contains results derived from the MIT altimetry data analysis. The results include the number of ARCDR ADF footprints used in computing the estimates of scattering properties for the pixel and estimates (and variances) of radius, rms surface tilt, and Fresnel reflectivity from the ARCDR.

(4) Stanford ALT Data Table
This table contains results from the Stanford analysis of altimetry data. Results include the number of SCVDR footprints used in computing the estimates of surface properties for this pixel, the centroid of the Doppler spectrum, the derived scattering function and the angles over which it is valid, variance of the individual points in the derived scattering function, and results of fitting analytic functions to the derived scattering function.

Ancillary Data
==============

Ancillary data for most processing at both MIT and Stanford was obtained from the data tapes and files received from the Magellan Project. These included trajectory and pointing information for the spacecraft, clock conversion tables, spacecraft engineering data, and SAR processing parameters. For calibration of the radar instrument itself, Magellan Project reports (including some received from Hughes Aircraft Co.[BARRY1987; CUEVAS1989; SE011]) were used. Documentation on handling of data at the Jet Propulsion Laboratory was also used [BRILL&MEISL1990; SCIEDR; SDPS101].

Coordinate System
=================

The data are presented in gridded formats, tiled to ensure that closely spaced points on the surface occupy nearby storage locations on the data storage medium. Four separate projections are used: sinusoidal equal area and Mercator for points within 89 degrees of the equator, and polar stereographic for points near the north and south poles. The projections are described by [SNYDER1987]; IAU conventions described by [DAVIESETAL1989] and Magellan Project assumptions [LYONS1988] have been adopted.

Software
========

A special library and several example programs are provided in source code form for reading the GVDR data files. The general-purpose example program will serve the needs of the casual user by accessing a given GVDR quantity over a specified region of GVDR pixels. More advanced users may want to write their own programs that use the GVDR library as a toolkit. The library, written in ANSI C, provides concise access methods for reading every quantity stored in the GVDR. It conveniently handles all geometric and tiling transformations and converts

```
  any compressed qualities to a standard native format. The general purpose program mentioned
  above provides an example of how to use this library.


Media/Format
============
  The GVDR will be delivered to the Magellan Project (or its successor) using compact disc
  write once (CD-WO) media. Formats will be based on standards for such products established by
  the Planetary Data System (PDS) [PDSSR1992]."

  CONFIDENCE_LEVEL_NOTE           = "


Confidence Level Overview
=========================
  The GVDR is intended to be the most systematic and comprehensive representation of the
  electromagnetic properties of the Venus surface that can be derived from Magellan data at
  this resolution. Nevertheless, there are limitations to what can be done with the data.


Review
======
  The GVDR will be reviewed internally by the Magellan Project prior to release to the
  planetary community. The GVDR will also be reviewed by PDS.


Data Coverage and Quality
=========================
  Because the orbit of Magellan was elliptical during most of its mapping operations, parts of
  the orbital coverage have higher resolution and higher signal-to-noise than others.

  Cycle 1 Mapping
  ---------------
    During Mapping Cycle 1, periapsis was near 10 degrees N latitude at altitudes of
    approximately 300 km over the surface. The altitude near the poles, on the other hand, was
    on the order of 3000 km. For all data types this means lower confidence in the results
    obtained at the poles than near the equator.

    Further, the spacecraft attitude was adjusted so that the SAR antenna was pointed at about
    45 degrees from nadir near periapsis; this was reduced to near 15 degrees at the poles.The
    objective was to compensate somewhat for the changing elevation and to provide scattering
    at higher incidence angles when the echo signal was expected to be strongest. The ALT
    antenna, at a constant 25 degree offset from the SAR antenna, followed in tandem but at
    angles which were not optimized for obtaining the best altimetry echo.

    During Mapping Cycle 1 almost half the orbits provided SAR images of the north pole;
    because of the orbit inclination, ALT data never extended beyond about 85N latitude in the
    north and 85S in the south. No SAR images of the south pole were acquired during Mapping
    Cycle 1 because the SAR antenna was always pointed to the left of the ground track; the
    Cycle 1SAR image strip near the south pole was at a latitude equator ward of 85S.

  Cycle 2 Mapping
  ---------------
    During much of Mapping Cycle 2, the spacecraft was flown ‘backwards’ so as to provide SAR
    images of the same terrain but with ‘opposite side’ illumination. This adjustment also
    meant that the SAR could image near the Venus south pole (but not near the north pole). The
    ALT data continued to be limited to latitudes equator ward of 85N and 85S.

  Cycle 3 Mapping
  ---------------
    During Mapping Cycle 3 the emphasis was on obtaining SAR data from the same side as in
    Cycle 1 but at different incidence angles (for radar stereo). In fact, most data were
    acquired at an incidence angle of about 25 degrees, which meant that the ALT antenna was
    usually aimed directly at nadir instead of drifting from side to side, as had been the case
    in Cycle 1.These Cycle 3 data, therefore, may be among the best from the altimeter. Dynamic
    range in SAR data was larger than in Cycle1 because the incidence angle was fixed rather
    than varying to compensate for the changing spacecraft height.

  All Cycles
  ----------
    It is important to remember that, since the SAR and ALT antennas were aimed at different
    parts of the planet during each orbit, building up a collection of composite scattering
    data for any single surface region requires that results from several orbits be integrated.
```

```
      In the case of data from polar regions, where only the SAR was able to probe, there will be
      no ALT data. When scheduling or other factors interrupted the systematic collection of
      data, there may be ALT data for some regions but no comparable SAR or radiometry data (or
      viceversa).

      Note that for all Cycles outages played an important role in determining coverage. For
      example, although a goal of Cycle 3 radar mapping was radar stereo, early orbits were used
      to collect data at nominal incidence angles that had been missed during Cycle 1 because of
      thermal problems with the spacecraft. A transmitter failure during Cycle 3 caused a loss of
      further data. It is not within the scope of this description to provide detailed
      information on data coverage.


  Limitations
  ===========
    Both the template fitting approach and the inversion approach will have their limitations in
    estimating overall surface properties for a region on Venus. The template calculation assumes
    that scattering is well-behaved at all incidence angles from 0 to 90 degrees and that a
    template representing that behavior can be constructed. The Hagfors function
    [HAGFORS1964]used by MIT, however, fails to give a finite rms surface tilt if used over this
    range of angles, so approximations based on a change in the scattering mechanism must be
    applied[HAGFORS&EVANS1968]. The inversion method [TYLER1992] is susceptible to noise at the
    higher incidence angles and this will corrupt solutions if not handled properly. Users of
    this data set should be aware that radar echoes are statistically variable and that each
    result has an uncertainty.

    A nominal nadir footprint can be assigned to altimetry results, but this footprint is biased
    near periapsis because the ALT antenna is rotated about 20 degrees from nadir (during Cycle
    1).Over polar regions in Cycle 1, the ALT antenna is rotated about 10 degrees to the opposite
    side of nadir. A more important consideration in polar regions is that the area illuminated
    by the ALT antenna is approximately 100 times as large as near periapsis because of the
    higher spacecraft altitude. The region contributing to echoes in polar regions -- and
    therefore the region over which estimates of Fresnel reflectivity and rms surface tilts apply
    -- is much larger than at periapsis."

  END_OBJECT                          = DATA_SET_INFORMATION

OBJECT                                = DATA_SET_MISSION
    MISSION_NAME                      = MAGELLAN
  END_OBJECT                          = DATA_SET_MISSION

  OBJECT                              = DATA_SET_TARGET
    TARGET_NAME                       = VENUS
  END_OBJECT                          = DATA_SET_TARGET

  OBJECT                              = DATA_SET_HOST
    INSTRUMENT_HOST_ID                = MGN
    INSTRUMENT_ID                     = RDRS
  END_OBJECT                          = DATA_SET_HOST

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                  = "BARRY1987"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
   REFERENCE_KEY_ID                   = "BRILL&MEISL1990"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
   REFERENCE_KEY_ID                   = "CUEVAS1989"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
   REFERENCE_KEY_ID                   = "DAVIESETAL1989"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
   REFERENCE_KEY_ID                   = "FORD1992"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION

  OBJECT                              = DATA_SET_REFERENCE_INFORMATION
   REFERENCE_KEY_ID                   = "FORD1993"
  END_OBJECT                          = DATA_SET_REFERENCE_INFORMATION
```

```
  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "FORD&PETTENGILL1992"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "HAGFORS1964"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "HAGFORS&EVANS1968"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "LYONS1988"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "PDSSR1992"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "PETTENGILLETAL1992"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "SCIEDR"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "SDPS101"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "SE011"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "SNYDER1987"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

  OBJECT                             = DATA_SET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                 = "TYLER1992"
  END_OBJECT                         = DATA_SET_REFERENCE_INFORMATION

END_OBJECT                           = DATA_SET
END
```

B.2 DATA_SET_COLL_ASSOC_DATA_SETS

The DATA_SET_COLL_ASSOC_DATA_SETS catalog object, a sub-object of the DATA_SET_COLLECTION object, is repeated for each data set associated with a DATA_SET_COLLECTION. For example, if there are three distinct data sets comprising a collection, this object will be repeated three different times – once for each data set.

B.2.1 Required Keywords

1. DATA_SET_ID

B.2.2 Optional Keywords

None

B.2.3 Required Objects

None

B.2.4 Optional Objects

None

B.2.5 Example

See the example of the DATA_SET_COLLECTION object in Section B.4.5.

B.3 DATA_SET_COLLECTION_REF_INFO

The DATA_SET_COLLECTION_REF_INFO catalog object, a sub-object of DATA_SET_COLLECTION object, associates a reference with a data set collection. It is repeated once for each reference identified in the DATA_SET_COLLECTION catalog object.

A separate REFERENCE catalog object must be completed to provide the associated citation for each reference.

B.3.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of “N/A”.

B.3.2 Optional Keywords

None

B.3.3 Required Objects

None

B.3.4 Optional Objects

None

B.3.5 Example

See the example for the DATA_SET_COLLECTION object in Section B.4.5.

B.4 DATA_SET_COLLECTION

The DATA_SET_COLLECTION catalog object is used to link several data sets as a collection to be used and distributed together.

B.4.1 Required Keywords

1. DATA_SET_COLLECTION_ID

B.4.2 Optional Keywords

None

B.4.3 Required Objects

1. DATA_SET_COLL_ASSOC_DATA_SETS
2. DATA_SET_COLLECTION_INFO
3. DATA_SET_COLLECTION_REF_INFO

B.4.4 Optional Objects

None

B.4.5 Usage Notes

One DATA_SET_COLLECTION_INFORMATION catalog object must be completed for each data set collection. One DATA_SET_COLLECTION_ASSOC_DATA_SETS catalog object must be completed for each data set associated with the data set collection. That is, if there is more than one data set, this object is repeated. Similarly, one DATA_SET_COLLECTION_REF_INFO catalog object is required for each individual reference associated with the data set collection. All references should be included that are relevant to providing more detailed / specific data set collection information; such as, description of the data set collection. These references may include published articles, books, papers, electronic publications, etc.

B.4.6 Example:

```
/* Template: Data Set Collection Template                    Rev: 1993-09-24       */

/* Note: Complete one template for each data set collection. Identify             */
/*       individual data sets that are included in the collection by              */
/*       repeating the 3 lines for the DATA_SET_COLL_ASSOC_DATA_SETS              */
/*       object. Identify each data set collection reference by                   */
/*       repeating the 3 lines for the DATA_SET_COLLECTION_REF_INFO object.       */
/*       Also complete a separate REFERENCE template for each new                 */
/*       reference submitted to PDS.                                              */

/* Hierarchy:   DATA_SET_COLLECTION                                               */
/*              DATA_SET_COLLECTION_INFO                                          */
/*              DATA_SET_COLL_ASSOC_DATA_SETS                                     */
/*              DATA_SET_COLLECTION_REF_INFO                                      */

OBJECT                                = DATA_SET_COLLECTION
  DATA_SET_COLLECTION_ID              = "PREMGN-E/L/H/M/V-4/5-RAD/GRAV-V1.0"

  OBJECT                              = DATA_SET_COLLECTION_INFO
    DATA_SET_COLLECTION_NAME          = "PRE-MGN E/L/H/M/V 4/5 RADAR/GRAVITY DATA V1.0"
    DATA_SETS                         = 15
    START_TIME                        = 1968-11-09T00:00:00
    STOP_TIME                         = 1988-07-27T00:00:00
    DATA_SET_COLLECTION_RELEASE_DT    = 1990-06-15
    PRODUCER_FULL_NAME                = "RAYMOND E. ARVIDSON"
    DATA_SET_COLLECTION_DESC          = "


  Data Set Collection Overview
  ============================
    This entity is a collection of selected Earth-based radar data of Venus, the Moon, Mercury,
    and Mars, Pioneer Venus radar data, airborne radar images of Earth, and line of sight
    acceleration data derived from tracking the Pioneer Venus Orbiter and Viking Orbiter 2.
    Included are 12.6 centimeter wavelength Arecibo Venus radar images, 12.6 to 12.9cm Goldstone
    Venus radar images and altimetry data, together with altimetry, brightness temperature,
    Fresnel reflectivity and rms slopes derived from the Pioneer Venus Radar Mapper. For the
    Moon, Haystack 3.8 centimeter radar images and Arecibo 12.6 and 70 centimeter radar images
    are included. Mars data include Goldstone altimetry data acquired between 1971 and 1982 and
    araster data set containing radar units that model Goldstone and Arecibo backscatter
    observations. Mercury data consist of Goldstone altimetry files. The terrestrial data were
    acquired over the Pisgah lava flows and the Kelso dune field in the Mojave Desert,
    California, and consist of multiple frequency, multiple incidence angle views of the same
    regions. Data set documentation is provided, with references that allow the reader to
    reconstruct processing histories. The entire data set collection and documentation are
    available on a CD-ROM entitled Pre-Magellan Radar and Gravity Data."

  DATA_SET_COLLECTION_USAGE_DESC =                                  "


  Data Set Collection Usage Overview
  ==================================
    The intent of the data set collection is to provide the planetary science community with
    radar and gravity data similar to the kinds of data that Magellan will begin collecting in
    the summer of 1990. The data set collection will be used for pre- Magellan analyses of Venus
    and for comparisons to actual Magellan data. The entire data set collection and documentation
    are available on a CD-ROM entitled Pre-Magellan Radar and Gravity Data. A list of the
    hardware and software that may be used to read this CD-ROM can be obtained from the PDS
    Geosciences Discipline Node."

  END_OBJECT                          = DATA_SET_COLLECTION_INFO

  OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                       = "NDC8-E-ASAR-4-RADAR-V1.0"
  END_OBJECT                          = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                       = "ARCB-L-RTLS-5-12.6CM-V1.0"
  END_OBJECT                          = DATA_SET_COLL_ASSOC_DATA_SETS
```

```
  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "ARCB-L-RTLS-4-70CM-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "ARCB-V-RTLS-4-12.6CM-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "ARCB-L-RTLS-3-70CM-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "GSSR-M-RTLS-5-ALT-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "GSSR-H-RTLS-4-ALT-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "GSSR-V-RTLS-5-12.6-9CM-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "HSTK-L-RTLS-4-3.8CM-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "ARCB/GSSR-M-RTLS-5-MODEL-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "P12-V-RSS-4-LOS-GRAVITY-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "P12-V-ORAD-4-ALT/RAD-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "P12-V-ORAD-5-RADAR-IMAGE-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "P12-V-ORAD-5-BACKSCATTER-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLL_ASSOC_DATA_SETS
    DATA_SET_ID                           = "VO2-M-RSS-4-LOS-GRAVITY-V1.0"
  END_OBJECT                              = DATA_SET_COLL_ASSOC_DATA_SETS

  OBJECT                                  = DATA_SET_COLLECTION_REF_INFO
    REFERENCE_KEY_ID                      = ARVIDSONETAL1990A
  END_OBJECT                              = DATA_SET_COLLECTION_REF_INFO

END_OBJECT                                = DATA_SET_COLLECTION
END
```

B.5 DATA_SET_COLLECTION_INFO

The DATA_SET_COLLECTION_INFO catalog object, a sub-object of DATA_SET_COLLECTION, provides an overview of content and usage, as well as other information specific to the data set collection. This object includes a free-form text description, DATA_SET_COLLECTION_DESC.

B.5.1 Required Keywords

1. DATA_SET_COLLECTION_DESC
2. DATA_SET_COLLECTION_NAME
3. DATA_SET_COLLECTION_RELEASE_DT
4. DATA_SET_COLLECTION_USAGE_DESC
5. DATA_SETS
6. PRODUCER_FULL_NAME
7. START_TIME
8. STOP_TIME

B.5.2 Optional Keywords

None

B.5.3 Required Objects

None

B.5.4 Optional Objects

None

B.5.5 Usage Notes

NOTE: The following paragraph headings and subheadings are recommended as the minimum set of headings needed to describe a data set collection adequately. Additional headings and sub-headings may be added as desired. Should any of the more common headings *not* appear within a text description, it will be considered not applicable to the data set collection.

B.5.5.1 DATA_SET_COLLECTION_INFO Headings

Data Set Collection Overview
A high-level description of the characteristics and properties of a data set collection

Data Set Collection Usage Overview
A high-level description of the intended use of a data set collection

B.5.6 Example

See the example of the DATA_SET_COLLECTION object in Section B.4.5.

B.6 DATA_SET_HOST

The DATA_SET_HOST catalog object, a sub-object of the DATA_SET catalog object, identifies one host/instrument pair associated with a data set.

B.6.1 Required Keywords

1. INSTRUMENT_HOST_ID
2. INSTRUMENT_ID

B.6.2 Optional Keywords

None

B.6.3 Required Objects

None

B.6.4 Optional Objects

None

B.6.5 Example

See the example for the DATA_SET object in Section B.1.6

B.7 DATA_SET_INFORMATION

The DATA_SET_INFORMATION catalog object, a sub-object of the DATA_SET catalog object, provides a high-level description of a single PDS data set.

B.7.1 Required Keywords

1. ABSTRACT_DESC
2. CITATION_DESC
3. CONFIDENCE_LEVEL_NOTE
4. DATA_OBJECT_TYPE
5. DATA_SET_COLLECTION_MEMBER_FLG
6. DATA_SET_DESC
7. DATA_SET_NAME
8. DATA_SET_RELEASE_DATE
9. DATA_SET_TERSE_DESC
10. DETAILED_CATALOG_FLAG
11. PRODUCER_FULL_NAME
12. START_TIME
13. STOP_TIME

B.7.2 Optional keywords

None

B.7.3 Required Objects

None

B.7.4 Optional Objects

None

B.7.5 Usage Notes

The DATA_SET_INFORMATION catalog object includes two free-form text description fields: DATA_SET_DESC and CONFIDENCE_LEVEL_NOTE. Following are recommended headings and subheadings for use in these fields, with brief descriptions of suggested contents.

Note: These headings and subheadings are recommended as the minimum set of headings needed to describe a data set adequately. Additional headings and sub-headings may be added as desired. Should any of the more common headings *not* appear within the description, they will be assumed to be not applicable to the data set.

B.7.5.1 DATA_SET_DESC Headings

Data Set Overview

A high level description of the characteristics and properties of a data set

Parameters

The primary parameters (measured or derived quantities) included in the data set, with units and sampling intervals

Processing

The overall processing used to produce the data set, including a description of the input data (and source), processing methods or software, and primary parameters or assumptions used to produce the data set

Data

Detailed description of each data type identified in the "Data Set Overview", (e.g., image data, table data, etc.)

Ancillary Data

Description of the ancillary information needed to interpret the data set. The ancillary information may or may not be provided along with the data set. If not, this description should include sources or references for locating the ancillary data.

Coordinate System

Description of the coordinate system(s) or frame(s) of reference to be used for proper interpretation of the data set

Software

Description of software relevant to the data, including software supplied with the data set as well as external software or systems that may be accessed independently to assist in visualization or analysis of the data

Media/Format

Description of the media on which the data set is delivered to PDS for distribution, including format information that may limit use of the data set on specific hardware platforms (e.g., binary/ASCII, IBM EBCDIC format)

B.7.5.2 CONFIDENCE_LEVEL_NOTE Headings

Confidence Level Overview

A high level description of the level of confidence (e.g., reliability, accuracy, or certainty) in the data

Review

Brief description of the review process that took place prior to release of the data set to insure the accuracy and completeness of the data and associated documentation

Data Coverage and Quality

Description of overall data coverage and quality. This section should include information about gaps in the data (both for times or regions) and details regarding how missing or poor data are flagged or filled, if applicable.

Limitations

Description of any limitations on the use of the data set. For example, discuss other data required to interpret the data properly, or any assumptions regarding special processing systems used to further reduce or analyze the data. If the data have been calibrated or otherwise corrected or derived, describe any known anomalies or uncertainties in the results.

B.7.5.3 CITATION_DESC Formation Rule

The CITATION_DESC keyword is subject to a formation rule described in detail in the CITATION_DESC element definition in the PDS Data Dictionary. A brief description is:

{ <author_name>, <author_name>, ... }, <data_set_name>, DATA_SET_ID,
NASA Planetary Data System, <year_of_peer_review>.

If a citation description is not defined, nor is applicable to the data set, the value "N/A" should be used.

B.7.5.4 OTHER - Data Supplier provided

Any other important information in addition to the headings above, as desired (e.g., data compression, time-tagging, etc.)

B.7.6 Example

See the example for the DATA_SET object in Section B.1.6.

B.8 DATA_SET_MAP_PROJECTION

The DATA_SET_MAP_PROJECTION catalog object is one of two distinct objects that together define the map projection used in creating the digital images in a PDS data set. The associated object that completes the definition is the IMAGE_MAP_PROJECTION, which is fully described in Appendix B.13 of this document.

The map projection information resides in these two objects essentially to reduce data redundancy and at the same time allow the inclusion of elements needed to process the data at the image level. Static information that is applicable to the complete data set resides in the DATA_SET_MAP_PROJECTION object while dynamic information that is applicable to the individual images resides in the IMAGE_MAP_PROJECTION object.

B.8.1 Required Keywords

1. DATA_SET_ID

B.8.2 Optional Keywords

None

B.8.3 Required Objects

1. DATA_SET_MAP_PROJECTION_INFO

B.8.4 Optional Objects

None

B.8.5 Example

```
PDS_VERSION_ID                          = PDS3
LABEL_REVISION_NOTE                     = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 80

SPACECRAFT_NAME                         = MAGELLAN
TARGET_NAME                             = VENUS

OBJECT                                  = DATA_SET_MAP_PROJECTION
  DATA_SET_ID                           = "MGN-V-RDRS-5-DIM-V1.0"

  OBJECT                                = DATA_SET_MAP_PROJECTION_INFO
    MAP_PROJECTION_TYPE                 = "SINUSOIDAL"
```

```
  MAP_PROJECTION_DESC                 = "


 Map Projection Overview
 =======================
  The FMAP (Magellan Full Resolution Radar Mosaic) is presented in a Sinusoidal Equal-Area map
  projection. In this projection, parallels of latitude are straight lines, with constant
  distances between equal latitude intervals. Lines of constant longitude on either side of the
  projection meridian are curved since longitude intervals decrease with the cosine of latitude
  to account for their convergence toward the poles. This projection offers a number of
  advantages for storing and managing global digital data; in particular, it is computationally
  simple, and data are stored in a compact form.

  The Sinusoidal Equal-Area projection is characterized by a projection longitude, which is the
  center meridian of the projection, and a scale, which is given in units of pixels/degree. The
  center latitude for all FMAP's is the equator. Each FMAP contains its own central meridian.
  The tiles that make up an FMAP all have the same central meridian as the FMAP.

  Lat/Lon, Line/Sample Transformations
  ------------------------------------
    The transformation from latitude and longitude to line and sample is given by the following
    equations:

    line = INT(LINE_PROJECTION_OFFSET - lat*MAP_RESOLUTION + 1.0)

    sample = INT(SAMPLE_PROJECTION_OFFSET - (lon -
             CENTER_LONGITUDE)*MAP_RESOLUTION*cos(lat) + 1.0)

    Note that integral values of line and sample correspond to center of a pixel. Lat and lon
    are the latitude and longitude of a given spot on the surface.

  Line Projection Offset
  ----------------------
    LINE_PROJECTION_OFFSET is the line number minus one on which the map projection origin
    occurs. The map projection origin is the intersection of the equator and the projection
    longitude. The value of LINE_PROJECTION_OFFSET is positive for images starting north of the
    equator and is negative for images starting south of the equator.

  Sample Projection Offset
  ------------------------
    SAMPLE_PROJECTION_OFFSET is the nearest sample number to the left of the projection
    longitude. The value of SAMPLE_PROJECTION_OFFSET is positive for images starting to the
    west of the projection longitude and is negative for images starting to the east of the
    projection longitude.

  Center Longitude
  ----------------
    CENTER_LONGITUDE is the value of the projection longitude,which is the longitude that
    passes through the center of the projection.

  The values for FMAP products will be 1408, 235, and 35.

  There are four PDS parameters that specify the latitude and longitude boundaries of an image.
  MAXIMUM_LATITUDE and MINIMUM_LATITUDE specify the latitude boundaries of the image,and
  EASTERNMOST_LONGITUDE and WESTERNMOST_LONGITUDE specify the longitudinal boundaries of the
  map.

  Definitions of other mapping parameters can be found in the Planetary Science Data
  Dictionary."

  ROTATIONAL_ELEMENT_DESC              = "See DAVIESETAL1989."

  OBJECT                              = DS_MAP_PROJECTION_REF_INFO
    REFERENCE_KEY_ID                  = "DAVIESETAL1989"
  END_OBJECT                          = DS_MAP_PROJECTION_REF_INFO

  OBJECT                              = DS_MAP_PROJECTION_REF_INFO
    REFERENCE_KEY_ID                  = "BATSON1987"
  END_OBJECT                          = DS_MAP_PROJECTION_REF_INFO

  OBJECT                              = DS_MAP_PROJECTION_REF_INFO
    REFERENCE_KEY_ID                  = "EDWARDS1987"
  END_OBJECT                          = DS_MAP_PROJECTION_REF_INFO
```

```
  END_OBJECT                                = DATA_SET_MAP_PROJECTION_INFO
END_OBJECT                                  = DATA_SET_MAP_PROJECTION

END
```

B.9 DATA_SET_MAP_PROJECTION_INFO

The DATA_SET_MAP_PROJECTION catalog object, a sub-object of DATA_SET_MAP_PROJECTION, defines the specific map projection of an image data set.

B.9.1 Required Keywords

1. MAP_PROJECTION_DESC
2. MAP_PROJECTION_TYPE
3. ROTATIONAL_ELEMENT_DESC

B.9.2 Optional Keywords

None

B.9.3 Required Objects

1. DS_MAP_PROJECTION_REF_INFO

B.9.4 Optional Objects

None

B.9.5 Usage notes

The MAP_PROJECTION_DESC text should contain at least one heading, "Map Projection Overview". This section should provide a description of the map projection of the data set, indicating mathematical expressions used for latitude/longitude or line/sample transformations, line and sample projection offsets, center longitudes, etc., as well as any assumptions made in processing. Subheadings may be used for each of these descriptions, if desired.

The ROTATIONAL_ELEMENT_DESC text should contain at least one heading, "Rotational Element Overview". This section should provide a description of the standard used for the definition of a planet's pole orientation and prime meridian, right ascension and declination, spin angle, etc. (Please see the *Planetary Science Data Dictionary* for complete description.) The value in this field may also be a bibliographic citation of a published work containing the rotation element description. In this case the "Rotational Element Overview" heading may be omitted.

An example of a DATA_SET_MAP_PROJECTION_INFO object may be found in the previous section detailing the DATA_SET_MAP_PROJECTION object.

B.9.6 Example

See the example for the DATA_SET_MAP_PROJECTION object in Section B.8.5.

B.10 DATA_SET_MISSION

The DATA_SET_MISSION object, a sub-object of DATA_SET catalog object, identifies an associated mission.

B.10.1 Required Keywords

1. MISSION_NAME

B.10.2 Optional Keywords

None

B.10.3 Required Objects

None

B.10.4 Optional Objects

None

B.10.5 Example

See the example for the DATA_SET object in Section B.1.6.

B.11 DATA_SET_REFERENCE_INFORMATION

The DATA_SET_REFERENCE_INFORMATION object, a sub-object of DATA_SET catalog object, is used to identify references relevant to a particular data set. . A separate object must be completed for each reference cited within the DATA_SET catalog object.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.11.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.11.2 Optional Keywords

None

B.11.3 Required Objects

None

B.11.4 Optional Objects

None

B.11.5 Example

See the example for the DATA_SET object in Section B.1.6.

B.12 DATA_SET_TARGET

The DATA_SET_TARGET object, a sub-object of DATA_SET catalog object, identifies an observed target.

B.12.1 Required Keywords

2. TARGET_NAME

B.12.2 Optional Keywords

None

B.12.3 Required Objects

None

B.12.4 Optional Objects

None

B.12.5 Example

See the example for the DATA_SET object in Section B.1.6.

B.13 DS_MAP_PROJECTION_REF_INFO

The DS_MAP_PROJECTION_REF_INFO object, a sub-object of DATA_SET_MAP_PROJECTION_INFO catalog object, is used to identify references relevant to a map projection. A separate object must be completed for each reference cited within the DATA_SET_MAP_PROJECTION_INFO catalog object.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.13.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.13.2 Optional Keywords

None

B.13.3 Required Objects

None

B.13.4 Optional Objects

None

B.13.5 Example

See the example for the DATA_SET_MAP_PROJECTION object in Section B.8.5.

B.14 IMAGE_MAP_PROJECTION

The IMAGE_MAP_PROJECTION object is one of two distinct objects that define the map projection used in creating cartographically registered digital images in a PDS data set. The other associated object that completes the definition is DATA_SET_MAP_PROJECTION (see Appendix B.8).

The map projection information resides in these two objects to reduce redundancy and at the same time to allow the inclusion of elements needed to process the data at the image level. Basically, static information that is applicable to the complete data set resides in the DATA_SET_MAP_PROJECTION object, while dynamic information that is applicable to the individual images resides in the IMAGE_MAP_PROJECTION object.

The LINE_FIRST_PIXEL, LINE_LAST_PIXEL, SAMPLE_FIRST_PIXEL, and SAMPLE_LAST_PIXEL keywords are used to indicate spatial orientation of a stored image. An image may have been shifted or flipped prior to being physically recorded. These keywords are used in calculating the mapping of pixels between the original image and the stored image.

The following equations give the byte offsets needed to determine the mapping of a pixel (X,Y) from the original image to a pixel in the stored image:

The sample offset from the first pixel is:

$$\frac{\text{SAMPLE\_BITS} * (\text{Y} - \text{SAMPLE\_FIRST\_PIXEL}) * \text{LINE\_SAMPLES}}{8 * (\text{SAMPLE\_LAST\_PIXEL} - \text{SAMPLE\_FIRST\_PIXEL} + 1)}$$

The line offset from the first image line is:

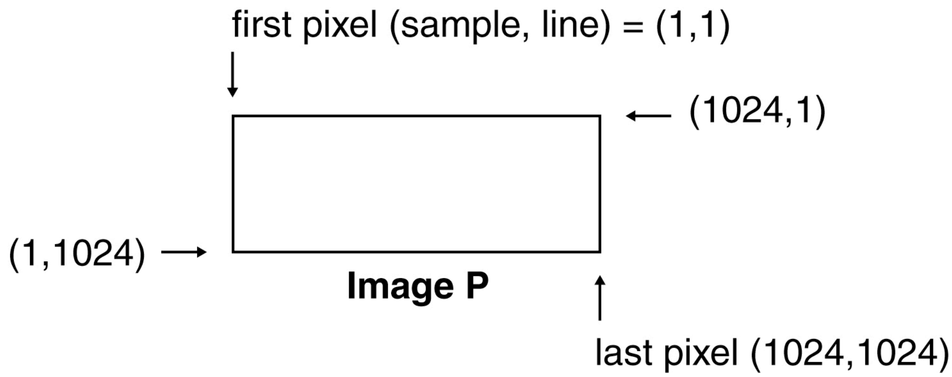
$$\frac{(\text{X} - \text{LINE\_FIRST\_PIXEL}) * \text{LINES}}{(\text{LINE\_LAST\_PIXEL} - \text{LINE\_FIRST\_PIXEL} + 1)}$$

Additionally, in any image, ABS (SAMPLE_LAST_PIXEL – SAMPLE_FIRST_PIXEL + 1) is always equal to LINE_SAMPLES, and ABS (LINE_LAST_PIXEL – LINE_FIRST_PIXEL + 1) is always equal to LINES.

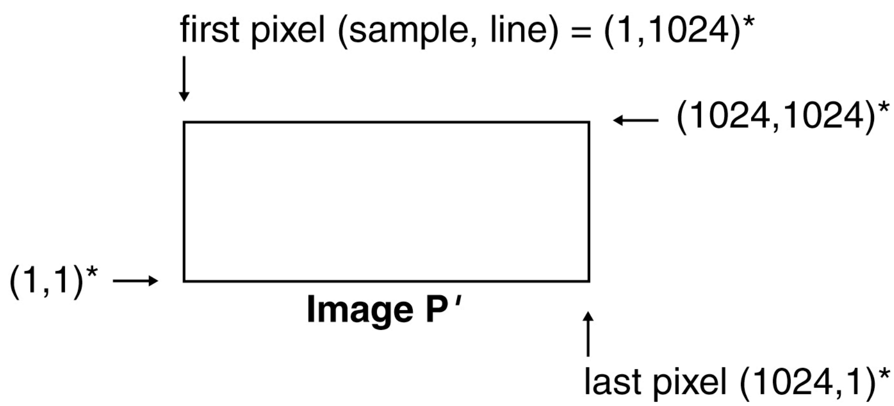
B.14.1 Example

Take a 1K by 1K 8-bit image which is rotated about the x-axis 180 degrees prior to being physically recorded.

Original Image: Positive direction is to the right and down



Stored Image: Positive direction is to the right and up



These pixel location values (*) are the positions from the original image. For example, the first pixel in the stored image (normally referred to as (1,1)) came from the position (1,1024) in the original image. These original values are used for the following IMAGE_MAP_PROJECTION keywords in the PDS label for the stored image:

```
SAMPLE_FIRST_PIXEL = 1
SAMPLE_LAST_PIXEL  = 1024
LINE_FIRST_PIXEL   = 1024
LINE_LAST_PIXEL    = 1
```

Now, given a pixel on the original image, P(X,Y) = (2,2), determine its location (P') in the stored image.

sample offset = (8 * (2 - 1) * 1024) / (8 * (1024 - 1 + 1)) = 1
line offset = ((2 - 1024) * 1024) / (1 - 1024 + 1) = (-1022)

Therefore, P' is located at (2, 1023) which is 1 byte from the first sample, and 1022 bytes (in the negative direction) from the first line in the stored image. See diagram above.

B.14.2 Required Keywords

1. MAP_PROJECTION_TYPE
2. A_AXIS_RADIUS
3. B_AXIS_RADIUS
4. C_AXIS_RADIUS
5. FIRST_STANDARD_PARALLEL
6. SECOND_STANDARD_PARALLEL
7. POSITIVE_LONGITUDE_DIRECTION
8. CENTER_LATITUDE
9. CENTER_LONGITUDE
10. REFERENCE_LATITUDE
11. REFERENCE_LONGITUDE
12. LINE_FIRST_PIXEL
13. LINE_LAST_PIXEL
14. SAMPLE_FIRST_PIXEL
15. SAMPLE_LAST_PIXEL
16. MAP_PROJECTION_ROTATION
17. MAP_RESOLUTION
18. MAP_SCALE
19. MAXIMUM_LATITUDE
20. MINIMUM_LATITUDE
21. EASTERNMOST_LONGITUDE
22. WESTERNMOST_LONGITUDE
23. LINE_PROJECTION_OFFSET
24. SAMPLE_PROJECTION_OFFSET
25. COORDINATE_SYSTEM_TYPE
26. COORDINATE_SYSTEM_NAME

B.14.3 Optional Keywords

1. DATA_SET_ID
2. IMAGE_ID
3. HORIZONTAL_FRAMELET_OFFSET
4. VERTICAL_FRAMELET_OFFSET
5. KEYWORD_LATITUDE_TYPE
6. OBLIQUE_PROJ_POLE_LATITUDE
7. OBLIQUE_PROJ_POLE_LONGITUDE
8. OBLIQUE_PROJ_POLE_ROTATION
9. OBLIQUE_PROJ_X_AXIS_VECTOR
10. OBLIQUE_PROJ_Y_AXIS_VECTOR
11. OBLIQUE_PROJ_Z_AXIS_VECTOR

B.14.4 Required Objects

1. DATA_SET_MAP_PROJECTION – This object is describe in Appendix B.

B.14.5 Optional Objects

None

B.14.6 Example

```
PDS_VERSION_ID                  = PDS3

/* File characteristics */
RECORD_TYPE                     = STREAM

/* Identification data elements */
DATA_SET_ID                     = "MGN-V-RDRS-5-GVDR-V1.0"
DATA_SET_NAME                   = "MAGELLAN VENUS RADAR SYSTEM GLOBAL
       DATA RECORD V1.0"
PRODUCT_ID                      = "IMP-NORTH.100"

MISSION_NAME                    = "MAGELLAN"
SPACECRAFT_NAME                 = "MAGELLAN"
INSTRUMENT_NAME                 = "VENUS"

ORBIT_START_NUMBER              = 376
ORBIT_STOP_NUMBER               = 4367
START_TIME                      = "N/A"
STOP_TIME                       = "N/A"
SPACECRAFT_CLOCK_START_COUNT    = "N/A"
SPACECRAFT_CLOCK_STOP_COUNT     = "N/A"

PRODUCT_CREATION_TIME           = 1994-05-07T22:09:27.000
PRODUCT_RELEASE_DATE            = 1994-05-13
PRODUCT_SEQUENCE_NUMBER         = 00000
PRODUCT_VERSION_TYPE            = "PRELIMINARY"

SOURCE_DATA_SET_ID              = {"MGN-V-RDRS-5-SCVDR-V1.0",
                                  "MGN-V-RDRS-CDR-ALT/RAD-V1.0"}
SOURCE_PRODUCT_ID               =
 {"SCVDR.00376-00399.1","SCVDR.00400-00499.1",
  "SCVDR.01100-01199.1","SCVDR.01200-01299.1","SCVDR.01300-01399.1",
  "SCVDR.01400-01499.1","SCVDR.01500-01599.1","SCVDR.01600-01699.1",
  "SCVDR.01700-01799.1","SCVDR.01800-01899.1","SCVDR.01900-01999.1",
  "ARCDRCD.001;2","ARCDRCD.002;1","ARCDRCD.003;1","ARCDRCD.004;1",
  "ARCDRCD.005;1","ARCDRCD.006;1","ARCDRCD.007;1","ARCDRCD.008;1",
  "ARCDRCD.017;1","ARCDRCD.018;1","ARCDRCD.019;1"}

SOFTWARE_FLAG                   = "Y"
```

```
PRODUCER_FULL_NAME              = "MICHAEL J. MAURER"
PRODUCER_INSTITUTION_NAME       = "STANFORD CENTER FOR RADAR ASTRONOMY"
PRODUCER_ID                     = "SCRA"
DESCRIPTION                     = "This file contains a single
  IMAGE_MAP_PROJECTION data object with an attached PDS label."

/* Data object definitions */
OBJECT                          = IMAGE_MAP_PROJECTION
  ^DATA_SET_MAP_PROJECTION      = "DSMAP.CAT"
  COORDINATE_SYSTEM_TYPE        = "BODY-FIXED ROTATING"
  COORDINATE_SYSTEM_NAME        = "PLANETOCENTRIC"
  MAP_PROJECTION_TYPE           = "STEREOGRAPHIC"
  A_AXIS_RADIUS                 = 6051.0 <KM>
  B_AXIS_RADIUS                 = 6051.0 <KM>
  C_AXIS_RADIUS                 = 6051.0 <KM>
  FIRST_STANDARD_PARALLEL       = "N/A"
  SECOND_STANDARD_PARALLEL      = "N/A"
  POSITIVE_LONGITUDE_DIRECTION  = "EAST"
  CENTER_LATITUDE               = 90
  CENTER_LONGITUDE              = 0
  REFERENCE_LATITUDE            = "N/A"
  REFERENCE_LONGITUDE           = "N/A"
  LINE_FIRST_PIXEL              = 1
  LINE_LAST_PIXEL               = 357
  SAMPLE_FIRST_PIXEL            = 1
  SAMPLE_LAST_PIXEL             = 357
  MAP_PROJECTION_ROTATION       = 0
  MAP_RESOLUTION                = 5.79478 <PIXEL/DEGREE>
  MAP_SCALE                     = 18.225 <KM/PIXEL>
  MAXIMUM_LATITUDE              = 90.00
  MINIMUM_LATITUDE              = 60.00
  EASTERNMOST_LONGITUDE         = 360.00
  WESTERNMOST_LONGITUDE         = 0.00
  LINE_PROJECTION_OFFSET        = 178
  SAMPLE_PROJECTION_OFFSET      = 178
END_OBJECT
```

B.15 INSTRUMENT

The INSTRUMENT catalog object is used to submit information about an instrument to PDS. Instruments are typically associated with a particular spacecraft or earth-based host, so the INSTRUMENT_HOST_ID keyword may identify either a valid SPACECRAFT_ID or EARTH_BASE_ID. (In those cases where a specific instrument was mounted on multiple earth-based hosts, the INSTRUMENT_HOST_ID may be multi-valued.) The catalog object includes a text description of the instrument and a sub-object for identifying reference information.

B.15.1 Required Keywords

1. INSTRUMENT_HOST_ID
2. INSTRUMENT_ID

B.15.2 Optional Keywords

None

B.15.3 Required Objects

1. INSTRUMENT_INFORMATION
2. INSTRUMENT_REFERENCE_INFO

B.15.4 Optional Objects

None

B.15.5 Usage Notes

One INSTRUMENT_INFORMATION catalog object must be completed for each instrument. An INSTRUMENT_REFERENCE_INFO catalog object is required for each individual reference associated with the instrument. All references should be included that are relevant to providing more detailed / specific instrument information; such as, description of the instrument, instrument documentation, review results, etc. These references may include published articles, books, papers, electronic publications, etc.

B.15.6 Example

```
/* Template: Instrument Template                           Rev: 1993-09-24      */

/* Note:Complete one template for each instrument. Identify each                */
/*      instrument reference by repeating the 3 lines for the                   */
/*      INSTRUMENT_REFERENCE_INFO object. Also complete a separate              */
/*      REFERENCE template for each new reference submitted to PDS.             */

/* Hierarchy:   INSTRUMENT                                                      */
/*              INSTRUMENT_INFORMATION                                          */
/*              INSTRUMENT_REFERENCE_INFO                                       */

PDS_VERSION_ID                         = PDS3
LABEL_REVISION_NOTE                    = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                            = STREAM

OBJECT                                 = INSTRUMENT
  INSTRUMENT_HOST_ID                   = "MGN"
  INSTRUMENT_ID                        = "RDRS"

  OBJECT                               = INSTRUMENT_INFORMATION
    INSTRUMENT_NAME                    = "RADAR SYSTEM"
    INSTRUMENT_TYPE                    = "RADAR"
    INSTRUMENT_DESC                    = "


  Instrument Overview
  ===================
    The Magellan radar system included a 3.7 m diameter high gain antenna (HGA) for SAR and
    radiometry and a smaller fan-beam antenna (ALTA) for altimetry. The system operated at 12.6
    cm wavelength. Common electronics were used in SAR, altimetry, and radiometry modes. The SAR
    operated in a burst mode; altimetry and radiometry observations were interleaved with the SAR
    bursts.

    Radiometry data were obtained by spending a portion of the time between SAR bursts and after
    altimeter operation in a passive (receive-only) mode, with the HGA antenna capturing the
    microwave thermal emission from the planet. Noise power within the 10-MHz receiver bandwidth
    was detected and accumulated for50 ms. To reduce the sensitivity to receiver gain changes in
    this mode, the receiver was connected on alternate bursts first to a comparison dummy load at
    a known physical temperature and then to the HGA. The short-term temperature resolution was
    about 2 K; the long-term absolute accuracy after calibration was about 20 K.

    The radar was manufactured by Hughes Aircraft Company and the 'build date' is taken to be
    1989-01-01. The radar dimensions were 0.304 by 1.35 by 0.902 (height by length by width in
    meters) and the mass was 126.1 kg.

    Instrument Id                                                : RDRS
    Instrument Host Id                                           : MGN
    Pi PDS User Id                                               : GPETTENGILL
    Instrument Name                                              : RADAR SYSTEM
    Instrument Type                                              : RADAR
    Build Date                                                   : 1989-01-01
    Instrument Mass                                              : 126.100000
    Instrument Length                                            : 1.350000
    Instrument Width                                             : 0.902000
    Instrument Height                                            : 0.304000
    Instrument Manufacturer Name                                 : HUGHES AIRCRAFT

    Platform Mounting Descriptions
    ------------------------------
      The spacecraft +Z axis vector was in the nominal direction of the HGA boresight. The +X
      axis vector was parallel to the nominal rotation axis of the solar panels. The +Y axis
      vector formed a right-handed coordinate system and was in the nominal direction of the star
      scanner boresight. The spacecraft velocity vector was in approximately the -Y direction
      when the spacecraft was oriented for left-looking SAR operation. The nominal HGA
      polarization was linear in the y-direction.

      Cone Offset Angle                :                     0.00
      Cross Cone Offset Angle          :                     0.00
      Twist Offset Angle               :                     0.00
```

The altimetry antenna boresight was in the x-z plane 25 degrees from the +Z direction and 65 degrees from the +X direction. The altimetry antenna was aimed approximately toward nadir during nominal radar operation. The altimetry antenna polarization was linear in the y-direction.

The medium gain antenna boresight was 70 degrees from the +Z direction and 20 degrees from the -Y direction. The low gain antenna was mounted on the back of the HGA feed; it's boresight was in the +Z direction and it had a hemispherical radiation pattern.

Principal Investigator

The Principal Investigator for the radar instrument was Gordon H. Pettengill.

For more information on the radar system see the papers by [JOHNSON1990] and [SAUNDERSETAL1990].

Scientific Objectives
=====================
See MISSION_OBJECTIVES_SUMMARY under MISSION.

Operational Considerations
==========================
The Magellan radar system was used to acquire radar back-scatter(SAR) images, altimetry, and radiometry when the spacecraft was close to the planet. Nominal operation extended from about 20minutes before periapsis until about 20 minutes after periapsis.In the SAR mode output from the radar receiver was sampled, blocks of samples were quantized using an adaptive procedure, and the results were stored on tape. In the altimetry mode samples were recorded directly, without quantization. Radiometry measurements were stored in the radar header records. During most of the remainder of each orbit, the HGA was pointed toward Earth and the contents of the tape recorder were transmitted to a station of the DSN at approximately 270 kilobits/second. SAR, altimetry, and radiometry data were then processed using ground software into images, altimetry profiles, estimates of backscatter coefficient, emissivity, and other quantities.

Calibration
===========
The radar was calibrated before flight using an active electronic target simulator [CUEVAS1989].

Operational Modes
=================
The Magellan radar system consisted of the following sections, each of which operated in the following modes:

Section Mode

SAR (SAR)	Synthetic Aperture Radar
ALT	Altimetry
RAD	Radiometry

(1) SAR Characteristics
In the Synthetic Aperture Radar mode, the radar transmitted bursts of phase-modulated pulses through its high gain antenna. Echo signals were captured by the antenna, simple dat the receiver output, and stored on tape after being quantized to reduce data volume. Pulse repetition rate and incidence angle were chosen to meet a minimum signal- to-noise ratio requirement (8 dB) for image pixels after ground processing. Multiple looks were used in processing to reduce speckle noise. Incidence angles varied from about 13 degree sat the pole to about 44 degrees at periapsis during normal mapping operations (e.g., Cycle 1); but other 'look angle profiles' were used during the mission.

Peak transmit power	: 350 watts
Transmitted pulse length	: 26.5 microsecs
Pulse repetition frequency	: 4400-5800 per sec
Time bandwidth product	: 60
Inverse baud width	: 2.26 MHz
Data quantization (I and Q)	: 2 bits each
Recorded data rate	: 750 kilobits/sec
Polarization (nominal)	: linear horizontal
HGA half-power full beam width	: 2.2 deg (azimuth)

```
                                                        : 2.5 deg (elev)
    one-way gain (from SAR RF port)                     : 35.7
    dBi System temperature (viewing Venus)              : 1250 K
    Surface resolution (range)                          : 120-360 m
    (along track)                                       : 120-150 m
    Number of looks                                     : 4 or more
    Swath width                                         : 25 km (approx)
    Antenna look angle                                  : 13-47 deg
    Incidence angle on surface                          : 18-50 deg

    Data Path Type                                      : RECORDED DATA
    PLAYBACK Instrument Power Consumption               : UNK

    (2) ALT Characteristics
    After SAR bursts (typically several times a second) groups of altimeter pulses were
    transmitted from a dedicated fan beam altimeter antenna (ALTA) directed toward the
    spacecraft's nadir. Output from the radar receiver was sampled, and the samples were stored
    on tape for transmission to Earth. During nominal left-looking SAR operation the ALTA
    pointed approximately 20 deg to the left of the spacecraft ground track at periapsis and
    about 10 deg to the right of the ground track near the north and south pole.

    Data quantization (I and Q)                         : 4 bits each
    Recorded data rate                                  : 35 kbs
    Polarization                                        : linear
    ALTA half-power full beam width
      (along track)                                     : 11 deg
    (cross track)                                       : 31 deg
    one-way gain referenced to ALT RF port              : 18.9
    dBi ALTA offset from HGA                            : 25 deg
    Burst interval                                      : 0.5-1.0 sec
    duration                                            : 1.0 millisec
    Dynamic range                                       : 30 dB (or more)

    Data Path Type                                      : RECORDED DATA
    PLAYBACK Instrument Power Consumption               : UNK

    (3) RAD Characteristics
    Radiometry measurements were made by the radar receiver and HGA in a receive-only mode that
    was activated after the altimetry mode to record the level of microwave radio thermale
    mission from the planet. Noise power within the 10-MHz receiver bandwidth was detected and
    accumulated for 50 ms. To reduce the sensitivity to receiver gain changes in this mode, the
    receiver was connected on alternate bursts first to a comparison dummy load at a known
    physical temperature and then to the HGA. The short-term temperature resolution was about
    2K; the long-term absolute accuracy after calibration was about20 K. At several times
    during the mission, radiometry measurements were carried out using known cosmic radio
    sources.

    Receiver Bandwidth                                  : 10 MHz
    Integration Time                                    : 50 millisecs
    Polarization (nominal)                              : linear horizontal
    Data Quantization                                   : 12 bits
    Data Rate                                           : 10-48 bits/sec
    HGA half-power full beam width                      : 2.2 deg
    System temperature (viewing Venus)                  : 1250 K
    Antenna look angle                                  : 13-47 deg
    Incidence angle on surface                          : 18-50 deg
    Surface resolution (along track)                    : 15-120 km
    (cross track)                                       : 20-125 km

    Data Path Type                                      : RECORDED DATA PLAYBACK
    Instrument Power Consumption                        : UNK "

  END_OBJECT                      = INSTRUMENT_INFORMATION

  OBJECT                         = INSTRUMENT_REFERENCE_INFO
    REFERENCE_KEY_ID             = "CUEVAS1989"
  END_OBJECT                     = INSTRUMENT_REFERENCE_INFO

  OBJECT                         = INSTRUMENT_REFERENCE_INFO
    REFERENCE_KEY_ID             = "JOHNSON1990"
  END_OBJECT                     = INSTRUMENT_REFERENCE_INFO

  OBJECT                         = INSTRUMENT_REFERENCE_INFO
```

```
    REFERENCE_KEY_ID                        = "SAUNDERSETAL1990"
  END_OBJECT                                = INSTRUMENT_REFERENCE_INFO

END_OBJECT                                  = INSTRUMENT

END
```

B.16 INSTRUMENT_HOST

The INSTRUMENT_HOST catalog object is used to describe a variety of instrument hosts, such as a spacecraft or an earth-based observatory.

B.16.1 Required Keywords

1. INSTRUMENT_HOST_ID

B.16.2 Optional Keywords

None

B.16.3 Required Objects

1. INSTRUMENT_HOST_INFORMATION
2. INSTRUMENT_HOST_REFERENCE_INFO

B.16.4 Optional Objects

None

B.16.5 Usage Notes

One INSTRUMENT_HOST_INFORMATION catalog object must be completed for each instrument host. An INSTRUMENT_HOST_REFERENCE_INFO catalog object is required for each individual reference associated with the instrument host. All references should be included that are relevant to providing more detailed / specific instrument host information; such as, description of the instrument host, instrument host documentation, review results, etc. These references may include published articles, books, papers, electronic publications, etc.

B.16.6 Example

```
/* Template: Instrument Host Template                              Rev: 1993-09-24       */

/* Note: Complete one template for each instrument host. Identify each                   */
/*       instrument host reference by repeating the 3 lines for the                      */
/*       INSTRUMENT_HOST_REFERENCE_INFO object. Also complete a separate                 */
/*       REFERENCE template for each new reference submitted to PDS.                     */

/* Hierarchy:   INSTRUMENT_HOST                                                          */
```

```
/*              INSTRUMENT_HOST_INFORMATION                                                */
/*              INSTRUMENT_HOST_REFERENCE_INFO                                             */

PDS_VERSION_ID                          = PDS3
LABEL_REVISION_NOTE                     = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                             = "STREAM"

OBJECT                                  = INSTRUMENT_HOST
  INSTRUMENT_HOST_ID                    = "MGN"

  OBJECT                                = INSTRUMENT_HOST_INFORMATION
    INSTRUMENT_HOST_NAME                = "MAGELLAN"
    INSTRUMENT_HOST_TYPE                = "SPACECRAFT"
    INSTRUMENT_HOST_DESC                = "

  Instrument Host Overview
  ========================
    The Magellan spacecraft was built by the Martin Marietta Corporation. The spacecraft
    structure included four major sections: High-Gain Antenna (HGA), Forward Equipment Module
    (FEM), Spacecraft Bus (including the solar array), and the Orbit Insertion Stage. Spacecraft
    subsystems included those for thermal control, power, attitude control, propulsion, command
    data and data storage, and telecommunications.

    The Magellan telecommunications subsystem contained all the hardware necessary to maintain
    communications between Earth and the spacecraft. The subsystem contained the radio frequency
    subsystem, the LGA, MGA, and HGA. The RFS performed the functions of carrier transponding,
    command detection and decoding, and telemetry modulation. The spacecraft was capable of
    simultaneous X-band and S-band uplink and downlink operations. The S-band operated at a
    transmitter power of 5 W, while the X-band operated at a power of 22 W. Uplink data rates
    were 31.25 and 62.5 bps (bits per second) with downlink data rates of 40 bps (emergency
    only), 1200 bps (real-time engineering rate), 115.2 kbps (kilobits per second) (radar down
    link backup), and 268.8 kbps (nominal).

    For more information on the Magellan spacecraft see the papers by [SAUNDERSETAL1990] and
    [SAUNDERSETAL1992]. "

  END_OBJECT                            = INSTRUMENT_HOST_INFORMATION

  OBJECT                                = INSTRUMENT_HOST_REFERENCE_INFO
    REFERENCE_KEY_ID                    = "SAUNDERSETAL1990"
  END_OBJECT                            = INSTRUMENT_HOST_REFERENCE_INFO

  OBJECT                                = INSTRUMENT_HOST_REFERENCE_INFO
    REFERENCE_KEY_ID                    = "SAUNDERSETAL1992"
  END_OBJECT                            = INSTRUMENT_HOST_REFERENCE_INFO

END_OBJECT                              = INSTRUMENT_HOST
END
```

B.17 INSTRUMENT_HOST_INFORMATION

The INSTRUMENT_HOST_INFORMATION object, a sub-object of the INSTRUMENT_HOST catalog object, provides a description of an instrument host. For spacecraft, this typically includes paragraphs on the various subsystems. Earth-based instrument host descriptions generally focus on geographic and facility elements.

B.17.1 Required Keywords

1. INSTRUMENT_HOST_DESC
2. INSTRUMENT_HOST_NAME
3. INSTRUMENT_HOST_TYPE

B.17.2 Optional Keywords

None

B.17.3 Required Objects

None

B.17.4 Optional Objects

None

B.17.5 Usage Notes

The INSTRUMENT_HOST_DESC keyword contains a text description of the spacecraft or ground observatory. It should contain at least one heading, “Instrument Host Overview”. This section should provide a high-level description of the characteristics and properties of the host. Other headings and sub-headings may be added as needed.

B.17.6 Example

See the example for the INSTRUMENT_HOST object in Section B.15.5.

B.18 INSTRUMENT_HOST_REFERENCE_INFO

The INSTRUMENT_HOST_REFERENCE_INFO object, a sub-object of the INSTRUMENT_HOST catalog object, associates a reference with an instrument host. A separate object must be completed for each reference cited within the INSTRUMENT_HOST catalog object.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.18.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.18.2 Optional Keywords

None

B.18.3 Required Objects

None

B.18.4 Optional Objects

None

B.18.5 Example

See the example for the INSTRUMENT_HOST object in Section B.15.5.

B.19 INSTRUMENT_INFORMATION

The INSTRUMENT_INFORMATION catalog object provides a description of the instrument.

B.19.1 Required Keywords

1. INSTRUMENT_DESC
2. INSTRUMENT_NAME
3. INSTRUMENT_TYPE

B.19.2 Optional Keywords

None

B.19.3 Required Objects

None

B.19.4 Optional Objects

None

B.19.5 Usage Notes

The following paragraph headings and suggested contents for the INSTRUMENT_DESC text are strongly recommended as the minimal set of information necessary to adequately describe an instrument. Additional headings may be appropriate for specific instruments and these also may be added here. Should any of the recommended headings *not* appear within the description, they will be considered not applicable to the data set.

Instrument Overview
A high-level description of the characteristics and properties of an instrument

Scientific Objectives
The scientific objectives of data obtained from this instrument

Calibration
Methods/procedures/schedules of instrument calibration (calibration stability, parameters, etc.)

<u>Operational Considerations</u>

Special circumstances or events that affect the instrument's ability to acquire high quality data , and which are reflected in the archive product. For example: spacecraft charging; thruster firings; contamination from other instruments; air quality; temperatures, etc.

<u>Detectors</u>

General description of the detector(s), including things like type of detector used, sensitivity and noise levels, detector fields of view, geometric factors, instrument/detector mounting descriptions (offset angles, pointing positions, etc.)

<u>Electronics</u>

Description of the instrument electronics and internal data processing (A-D converter)

<u>Filters</u>

Description of instrument filters and filter calibrations (filter type, center wavelength, min/max wavelength), as applicable

<u>Optics</u>

Description of instrument optics (focal lengths, transmittance, diameter, resolution, t_number, etc.), as applicable

<u>Location</u>

Latitude and longitude location, for earth-based instruments

<u>Operational Modes</u>

Description of instrument configurations for data acquisitions. Description of "modes" (scan, gain, etc.) of data acquisition and of measured parameter(s) and/or data sampling rates or schemes used in each mode

<u>Subsystems</u>

Logical subsystems of the instrument, including descriptions of each subsystem, how it's used, which "modes" make use of which subsystem, etc.

<u>Measured Parameters</u>

Description of what the instrument measures directly (particle counts, magnetic field components, radiance, current/voltage ratios, etc.), plus descriptions and definitions of these measurements (min/max, noise levels, units, time interval between measurements, etc.)

OTHER - Data Supplier provided: Any other important information in additional headings as desired (e.g., data reduction, data compression, time-tagging, diagnostics, etc.)

B.19.6 Example

See the example for the INSTRUMENT object in Section B.14.5.

B.20 INSTRUMENT_REFERENCE_INFO

The INSTRUMENT_REFERENCE_INFO catalog object associates a reference with an instrument description. A separate object must be completed for each reference cited within the INSTRUMENT catalog object. Include any important references such as instrument description and calibration documents. These can be published articles, internal documents or informal memoranda.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.20.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.20.2 Optional Keywords

None

B.20.3 Required Objects

None

B.20.4 Optional Objects

None

B.20.5 Example

See the example for the INSTRUMENT object in Section B.14.5.

B.21 INVENTORY

One INVENTORY catalog object is completed for each node responsible for orderable data sets from the PDS catalog. This object provides the inventory information necessary to facilitate the ordering of these data sets.

B.21.1 Required Keywords

1. NODE_ID

B.21.2 Optional Keywords

None

B.21.3 Required Objects

1. INVENTORY_DATA_SET_INFO

B.21.4 Optional Objects

None

B.21.5 Example

```
/* Template: InventoryTemplate                          Rev: 1990-03-20                    */

/* Note: The INVENTORY template shall be completed once for each node that is responsible     */
/*  for orderable data sets from the PDS catalog. The following hierarchy of templates provide */
/*  the necessary inventory information which will facilitate the ordering of these data sets. */

/* Hierarchy:   INVENTORY                                                            */
/*              INVENTORY_DATA_SET_INFO                                              */
/*              INVENTORY_NODE_MEDIA_INFO                                            */

OBJECT                                  = INVENTORY
  NODE_ID                               = "IMAGING"

  OBJECT                                = INVENTORY_DATA_SET_INFO
    PRODUCT_DATA_SET_ID                 = "VG2-N-ISS-2-EDR-V1.0"

    OBJECT                              = INVENTORY_NODE_MEDIA_INFO
      MEDIUM_TYPE                       = "MAG TAPE"
      MEDIUM_DESC                       = "INDUSTRY STD 1/2IN;1600 OR 6250 BPI"
      COPIES                            = 1
      INVENTORY_SPECIAL_ORDER_NOTE      = "Not applicable."
    END_OBJECT                          = INVENTORY_NODE_MEDIA_INFO

    OBJECT                              = INVENTORY_NODE_MEDIA_INFO
```

```
      MEDIUM_TYPE                         = "CD-ROM"
      MEDIUM_DESC                         = "Compact Disk"
      COPIES                              = 1
      INVENTORY_SPECIAL_ORDER_NOTE        = "Not applicable."
    END_OBJECT                            = INVENTORY_NODE_MEDIA_INFO

  END_OBJECT                              = INVENTORY_DATA_SET_INFO
END_OBJECT                                = INVENTORY


OBJECT                                    = INVENTORY
  NODE_ID                                 = "NSSDC"

  OBJECT                                  = INVENTORY_DATA_SET_INFO
    PRODUCT_DATA_SET_ID                   = "VG2-N-ISS-2-EDR-V1.0"

    OBJECT                                = INVENTORY_NODE_MEDIA_INFO
      MEDIUM_TYPE                         = "CD-ROM"
      MEDIUM_DESC                         = "Compact Disk"
      COPIES                              = 1
      INVENTORY_SPECIAL_ORDER_NOTE        = "Not applicable."
    END_OBJECT                            = INVENTORY_NODE_MEDIA_INFO

  END_OBJECT                              = INVENTORY_DATA_SET_INFO
END_OBJECT                                = INVENTORY
END
```

B.22 INVENTORY_DATA_SET_INFO

The INVENTORY_DATA_SET_INFO object, sub-object of the INVENTORY catalog object, identifies a data set through the DATA_SET_ID. This object is repeated once for each orderable and cataloged PDS data set.

B.22.1 Required Keywords

1. PRODUCT_DATA_SET_ID

B.22.2 Optional Keywords

None

B.22.3 Required Objects

1. INVENTORY_NODE_MEDIA_INFO

B.22.4 Optional Objects

None

B.22.5 Example

See the example for the INVENTORY object in Section B.20.5.

B.23 INVENTORY_NODE_MEDIA_INFO

The INVENTORY_NODE_MEDIA_INFO object, a sub-object of the INVENTORY_DATA_SET_INFO object, provides information about a data set's distribution medium. This object is repeated for each type of distribution medium.

B.23.1 Required Keywords

1. COPIES
2. INVENTORY_SPECIAL_ORDER_NOTE
3. MEDIUM_DESC
4. MEDIUM_TYPE

B.23.2 Optional Keywords

None

B.23.3 Required Objects

None

B.23.4 Optional Objects

None

B.23.5 Example

See the example for the INVENTORY object in Section B.20.5.

B.24 MISSION

The MISSION catalog object is used to submit information about a mission or observing campaign to PDS. Sub-objects are included for identifying associated instrument hosts, targets, and references.

B.24.1 Required Keywords

1. MISSION_NAME

B.24.2 Optional Keywords

None

B.24.3 Required Objects

1. MISSION_HOST
2. MISSION_INFORMATION
3. MISSION_REFERENCE_INFORMATION

B.24.4 Optional Objects

None

B.24.5 Usage Notes

One MISSION_INFORMATION catalog object must be completed for each mission. A MISSION_HOST catalog object must be completed for each mission host associated with the mission, and one MISSION_REFERENCE_INFORMATION catalog object is required for each individual reference associated with the mission (e.g., articles, papers, memoranda, published data, etc.). All references should be included that are relevant to providing more detailed / specific mission information; such as, description of the mission, phases of the mission, mission objectives, mission documentation, review results, etc. These references may include published articles, books, papers, electronic publications, etc.

B.24.6 Example

```
/* Template: Mission Template                         Rev: 1993-09-24           */

/* Note:Complete one template for each mission or campaign. Identify            */
/*      multiple hosts associated with the mission by repeating the             */
/*      lines beginning and ending with the MISSION_HOST values. For            */
/*      each instrument_host identified, repeat the 3 lines for the             */
/*      MISSION_TARGET object for each target associated with the host.         */
/*      Also complete a separate REFERENCE template for each new                */
/*      reference submitted to PDS.                                             */

/* Hierarchy: MISSION                                                           */
/*            MISSION_INFORMATION                                               */
/*            MISSION_HOST                                                      */
/*            MISSION_TARGET                                                    */
/*            MISSION_REFERENCE_INFORMATION                                     */

PDS_VERSION_ID                      = PDS3
LABEL_REVISION_NOTE                 = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                         = STREAM

OBJECT                              = MISSION
  MISSION_NAME                      = "MAGELLAN"

  OBJECT                            = MISSION_INFORMATION
    MISSION_START_DATE              = 1989-05-04
    MISSION_STOP_DATE               = UNK
    MISSION_ALIAS_NAME              = { "Venus Radar Mapper (VRM)", "SP-18" }
    MISSION_DESC                    = "


  Mission Overview
  ================
    The Magellan spacecraft was launched from the Kennedy Space Center on 4 May 1989. The
    spacecraft was deployed from the Shuttle cargo bay after the Shuttle achieved parking
    orbit.Magellan, using an inertial upper stage rocket, was then placed into a Type IV transfer
    orbit to Venus where it carried out radar mapping and gravity studies starting in August
    1990. The Mission has been described in many papers including two special issues of the
    Journal of Geophysical Research [VRMPP1983;SAUNDERSETAL1990; JGRMGN1992]. The radar system is
    also described in [JOHNSON1990].

    The aerobraking phase of the mission was designed to change the Magellan orbit from eccentric
    to nearly circular. This was accomplished by dropping periapsis to less than 150 km above the
    surface and using atmospheric drag to reduce the energy in the orbit. Aerobraking ended on 3
    August 1993, and periapsis was boosted above the atmosphere leaving the spacecraft in an
    orbit that was 540 km above the surface at apoapsis and 197 km above the surface at
    periapsis. The orbit period was 94 minutes. The spacecraft remained on its medium-gain
    antenna in this orbit until Cycle 5 began officially on 16 August 1993.

    During Cycles 5 and 6 the orbit was low and approximately circular. The emphasis was on
    collecting high-resolution gravity data. Two bistatic surface scattering experiments were
    conducted, one on 6 October (orbits 9331, 9335, and 9336) and the second on 9 November
    (orbits 9846-9848).


  Mission Phases
  ==============
    Mission phases were defined for significant spacecraft activity periods. During orbital
    operations a ‘cycle’ was approximately the time required for Venus to rotate once under the
    spacecraft (about 243 days). But there were orbit adjustments and other activities that made
    some mapping cycles not strictly contiguous and slightly longer or shorter than the rotation
    period.

    PRELAUNCH
    ---------
      The prelaunch phase extended from delivery of the spacecraft to Kennedy Space Center until
      the start of the launch countdown.

      Spacecraft Id                                              : MGN
      Target Name                                                : VENUS
      Mission Phase Start Time                                   : 1988-09-01
      Mission Phase Stop Time                                    : 1989-05-04
      Spacecraft Operations Type                                 : ORBITER
```

```
LAUNCH
------
  The launch phase extended from the start of launch countdown until completion of the
  injection into the Earth- Venus trajectory.

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1989-05-04
  Mission Phase Stop Time                                    : 1989-05-04
  Spacecraft Operations Type                                 : ORBITER

CRUISE
------
  The cruise phase extended from injection into the Earth-Venus trajectory until 10 days
  before Venus orbit insertion.

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1989-05-04
  Mission Phase Stop Time                                    : 1990-08-01
  Spacecraft Operations Type                                 : ORBITER

ORBIT INSERTION
---------------
  The Venus orbit insertion phase extended from 10 days before Venus orbit insertion until
  burnout of the solid rocket injection motor.

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1990-08-01
  Mission Phase Stop Time                                    : 1990-08-10
  Spacecraft Operations Type                                 : ORBITER

ORBIT CHECKOUT
--------------
  The orbit trim and checkout phase extended from burnout of the solid rocket injection motor
  until the beginning of radar mapping.

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1990-08-10
  Mission Phase Stop Time                                    : 1990-09-15
  Spacecraft Operations Type                                 : ORBITER

MAPPING CYCLE 1
---------------
  The first mapping cycle extended from completion of the orbit trim and checkout phase until
  completion of one cycle of radar mapping (approximately 243 days).

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1990-09-15
  Mission Phase Stop Time                                    : 1991-05-15
  Spacecraft Operations Type                                 : ORBITER

MAPPING CYCLE 2
---------------
  The second mapping cycle extended from completion of the first mapping cycle through an
  additional cycle of mapping.Acquisition of 'right-looking' SAR data was emphasized. Radio
  occultation measurements were carried out on orbits 3212-3214.A period of battery
  reconditioning followed completion of Cycle 2.

  Spacecraft Id                                              : MGN
  Target Name                                                : VENUS
  Mission Phase Start Time                                   : 1991-05-16
  Mission Phase Stop Time                                    : 1992-01-17
  Spacecraft Operations Type                                 : ORBITER

MAPPING CYCLE 3
---------------
  The third mapping cycle extended from completion of battery reconditioning through an
  additional cycle of mapping (approximately 243 days). Acquisition of 'stereo' SAR data was
  emphasized. The last orbit in the third cycle was orbit5747.
```

```
    Spacecraft Id                                              : MGN
    Target Name                                                : VENUS
    Mission Phase Start Time                                   : 1992-01-24
    Mission Phase Stop Time                                    : 1992-09-14
    Spacecraft Operations Type                                 : ORBITER

  MAPPING CYCLE 4
  ---------------
    The fourth mapping cycle extended from completion of the third mapping cycle through an
    additional cycle of mapping. Acquisition of radio tracking data for gravity studies was
    emphasized. Radio occultation measurements were carried out on orbits 6369, 6370, 6471, and
    6472. Because of poor observing geometry for gravity data collection at the beginning of
    the cycle, this cycle was extended 10 days beyond the nominal 243 days. Orbits included
    within the fourth cycle were 5748 through 7626. Periapsis was lowered on orbit 5752to
    improve sensitivity to gravity features in Cycle 4.

    Spacecraft Id                                              : MGN
    Target Name                                                : VENUS
    Mission Phase Start Time                                   : 1992-09-14
    Mission Phase Stop Time                                    : 1993-05-25
    Spacecraft Operations Type                                 : ORBITER

  AEROBRAKING
  -----------
    The aerobraking phase extended from completion of the fourth mapping cycle through
    achievement of a near- circular orbit.Circularization was achieved more quickly than
    expected; the first gravity data collection in the circular orbit was not scheduled until
    11 days later. Orbits included within the aerobraking phase were 7627 through 8392.

    Spacecraft Id                                              : MGN
    Target Name                                                : VENUS
    Mission Phase Start Time                                   : 1993-05-26
    Mission Phase Stop Time                                    : 1993-08-05
    Spacecraft Operations Type                                 : ORBITER

  MAPPING CYCLE 5
  ---------------
    The fifth mapping cycle extended from completion of the aerobraking phase through an
    additional cycle of mapping (approximately 243 days). Acquisition of radio tracking data
    for gravity studies was emphasized. The first orbit in the fifth cycle was orbit 8393.

    Spacecraft Id                                              : MGN
    Target Name                                                : VENUS
    Mission Phase Start Time                                   : 1993-08-16
    Mission Phase Stop Time                                    : 1994-04-15
    Spacecraft Operations Type                                 : ORBITER

  MAPPING CYCLE 6
  ---------------
    The sixth mapping cycle extended from completion of the fifth mapping cycle through an
    additional cycle of mapping (approximately 243 days). Acquisition of radio tracking data
    for gravity studies was emphasized. The first orbit in the sixth cycle was orbit 12249.

    Spacecraft Id                                              : MGN
    Target Name                                                : VENUS
    Mission Phase Start Time                                   : 1994-04-16
    Mission Phase Stop Time                                    : TBD
    Spacecraft Operations Type                                 : ORBITER"

  MISSION_OBJECTIVES_SUMMARY      = "


Mission Objectives Overview
===========================

  Volcanic and Tectonic Processes
  -------------------------------
    Magellan images of the Venus surface show widespread evidence for volcanic activity. A
    major goal of the Magellan mission was to provide a detailed global characterization of
    volcanic land forms on Venus and an understanding of the mechanics of volcanism in the
    Venus context. Of particular interest was the role of volcanism in transporting heat
    through the lithosphere.While this goal will largely be accomplished by a careful analysis
```

```
    of images of volcanic features and of the geological relationships of these features to
    tectonic and impact structures, an essential aspect of characterization will be an
    integration of image data with altimetry and other measurements of surface properties....

    For more information on volcanic and tectonic investigations see papers by [HEADETAL1992]
    and [SOLOMONETAL1992], respectively.

  Impact Processes
  ----------------
    The final physical form of an impact crater has meaning only when the effects of the
    cratering event and any subsequent modification of the crater can be distinguished. To this
    end, a careful search of the SAR images can identify and characterize both relatively
    pristine and degraded impact craters, together with their ejecta deposits (in each size
    range) as well as distinguishing impact craters from those of volcanic origin.The
    topographic measures of depth-to-diameter ratio, ejecta thickness distribution as a
    function of distance from the crater, and the relief of central peaks contribute to this
    documentation.

    For more information on investigations of impact processes see[SCHABERETAL1992].

  Erosional, Depositional, and Chemical Processes
  -----------------------------------------------
    The nature of erosional and depositional processes on Venus is poorly known, primarily
    because the diagnostic landforms typically occur at a scale too small to have been resolved
    in Earth-based or Venera 15/16 radar images. Magellan images show wind eroded terrains,
    landforms produced by deposition (dunefields), possible landslides and other down slope
    movements, as well as aeolian features such as radar bright or dark streaks 'downwind' from
    prominent topographic anomalies. One measure of weathering, erosion, and deposition is
    provided by the extent to which soil covers the surface (for Venus, the term soil is used
    for porous material, as implied by its relatively low value of bulk dielectric constant).
    The existence of such material, and its dependence on elevation and geologic setting,
    provide important insights into the interactions that have taken place between the
    atmosphere and the lithosphere.

    For more information on erosional, depositional, and chemical processes see papers by
    [ARVIDSONETAL1992], [GREELEYETAL1992],and [GREELEYETAL1994].

  Isostatic and Convective Processes
  ----------------------------------
    Topography and gravity are intimately and inextricably related, and must be jointly
    examined when undertaking geophysical investigations of the interior of a planet, where
    isostatic and convective processes dominate. Topography provides a surface boundary
    condition for modeling the interior density of Venus.

    For more information on topography and gravity see papers by[FORD&PETTENGILL1992],
    [KONOPLIVETAL1993], and[MCNAMEEETAL1993]. "

END_OBJECT                             = MISSION_INFORMATION

OBJECT                                 = MISSION_HOST
  INSTRUMENT_HOST_ID                   = "MGN"

  OBJECT                               = MISSION_TARGET
    TARGET_NAME                        = "VENUS"
  END_OBJECT                           = MISSION_TARGET
END_OBJECT                             = MISSION_HOST

OBJECT                                 = MISSION_REFERENCE_INFORMATION
  REFERENCE_KEY_ID                     = "ARVIDSON1991"
END_OBJECT                             = MISSION_REFERENCE_INFORMATION

OBJECT                                 = MISSION_REFERENCE_INFORMATION
  REFERENCE_KEY_ID                     = "ARVIDSONETAL1992"
END_OBJECT                             = MISSION_REFERENCE_INFORMATION

OBJECT                                 = MISSION_REFERENCE_INFORMATION
  REFERENCE_KEY_ID                     = "CAMPBELLETAL1992"
END_OBJECT                             = MISSION_REFERENCE_INFORMATION

...

OBJECT                                 = MISSION_REFERENCE_INFORMATION
  REFERENCE_KEY_ID                     = "TYLER1992"
```

```
  END_OBJECT                              = MISSION_REFERENCE_INFORMATION

  OBJECT                                  = MISSION_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                      = "VRMPP1983"
  END_OBJECT                              = MISSION_REFERENCE_INFORMATION

END_OBJECT                                = MISSION
END
```

B.25 MISSION_HOST

The MISSION_HOST object, a sub-object of the MISSION catalog object, is completed for each instrument host associated with a mission or observing campaign. If there is more than one instrument host involved in the mission, this object is repeated.

B.25.1 Required Keywords

1. INSTRUMENT_HOST_ID

B.25.2 Optional Keywords

None

B.25.3 Required Objects

1. MISSION_TARGET

B.25.4 Optional Objects

None

B.25.5 Example

See the example for the MISSION object in Section B.23.5.

B.26 MISSION_INFORMATION

The MISSION_INFORMATION object, a sub-object of the MISSION catalog object, provides start and stop times and text descriptions of a mission (or observing campaign) and its objectives. Suggested content includes agency involvement, spacecraft/observatory utilized, mission scenario including phases, technology and scientific objectives.

B.26.1 Required Keywords

1. MISSION_ALIAS_NAME
2. MISSION_DESC
3. MISSION_OBJECTIVES_SUMMARY
4. MISSION_START_DATE
5. MISSION_STOP_DATE

B.26.2 Optional Keywords

None

B.26.3 Required Objects

None

B.26.4 Optional Objects

None

B.26.5 Usage notes

The following paragraph headings and suggested contents for the MISSION_DESC and MISSION_OBJECTIVES_SUMMARY text are strongly recommended as the minimal set of information necessary to adequately describe a mission or observing campaign. Additional headings may be added as needed.

B.26.5.1 MISSION_DESC Headings

Mission Overview

A high-level description of a mission

Mission Phases

A description of each phase of a mission, starting with the pre-launch phase and continuing through end-of-mission, including: start and stop times of each phase; intended operations; targets; and mission phase objectives

B.26.5.2 MISSION_OBJECTIVES_SUMMARY Headings

Mission Objectives Overview

A high-level description of the objectives of the mission

B.26.6 Example

See the example for the MISSION object in Section B.23.5.

B.27 MISSION_REFERENCE_INFORMATION

The MISSION_REFERENCE_INFORMATION object, a sub-object of the MISSION catalog object, associates a reference with a mission. A separate object must be completed for each reference cited within the MISSION catalog object.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.27.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.27.2 Optional Keywords

None

B.27.3 Required Objects

None

B.27.4 Optional Objects

None

B.27.5 Example

See the example for the MISSION object in Section B.23.5.

B.28 MISSION_TARGET

The MISSION_TARGET object, a sub-object of the MISSION_HOST catalog object, associates a target with a mission host. One MISSION_TARGET catalog object is completed for each target associated with a mission host.

B.28.1 Required Keywords

1. TARGET_NAME

B.28.2 Optional Keywords

None

B.28.3 Required Objects

None

B.28.4 Optional Objects

None

B.28.5 Example

See the example for the MISSION object in Section B.23.5.

B.29 PERSONNEL

The PERSONNEL catalog object is used to provide new or updated information for personnel associated with PDS in some capacity. Associated personnel include data suppliers and producers for data sets or volumes archived with PDS, as well as PDS node personnel and contacts within other agencies and institutions.

B.29.1 Required Keywords

1. PDS_USER_ID

Note: With respect to new submissions, contact a PDS Data Engineer to obtain a valid and unique PDS_USER_ID. The value is constructed using the initial of the first name and the last name (e.g., John Smith would become PDS_USER_ID = “JSMITH”). The Data Engineer will ensure that the newly constructed value does not conflict with a previous entry in the catalog.

B.29.2 Optional Keywords

None

B.29.3 Required Objects

1. PERSONNEL_ELECTRONIC_MAIL
2. PERSONNEL_INFORMATION

B.29.4 Optional Objects

None

B.29.5 Usage Notes

One PERSONNEL_INFORMATION catalog object must be completed for each person. One PERSONNEL_ELECTRONIC_MAIL catalog object must be completed for each email address associated with the person. That is, if there is more than one email address, this object is repeated.

B.29.6 Example

```
/* Template: Personnel Template                         Rev: 1993-09-24       */

/* Note:Complete one for each new PDS user, data supplier, or data            */
/*producer. If more than one electronic mail address is available             */
/*repeat the lines for the PERSONNEL_ELECTRONIC_MAIL object.                  */

/* Hierarchy:   PERSONNEL                                                     */
/*              PERSONNEL_INFORMATION                                         */
/*              PERSONNEL_ELECTRONIC_MAIL                                     */

PDS_VERSION_ID                        = PDS3
LABEL_REVISION_NOTE                   = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                           = STREAM

OBJECT                                = PERSONNEL
  PDS_USER_ID                         = PFORD

  OBJECT                              = PERSONNEL_INFORMATION
    FULL_NAME                         = "PETER G. FORD"
    LAST_NAME                         = FORD
    TELEPHONE_NUMBER                  = "6172536485"
    ALTERNATE_TELEPHONE_NUMBER        = "6172534287"
    FAX_NUMBER                        = "6172530861"
    INSTITUTION_NAME                  = "MASSACHUSETTS INSTITUTE OF TECHNOLOGY"
    NODE_ID                           = "GEOSCIENCE"
    PDS_AFFILIATION                   = "NODE OPERATIONS MANAGER"
    PDS_ADDRESS_BOOK_FLAG             = Y
    REGISTRATION_DATE                 = 1990-02-06
    ADDRESS_TEXT                      = "Massachusetts Institute of Technology
                                         Center for Space Research Building 37-601
                                         Cambridge, MA 02139"
  END_OBJECT                          = PERSONNEL_INFORMATION

  OBJECT                              = PERSONNEL_ELECTRONIC_MAIL
    ELECTRONIC_MAIL_ID                = "PGF@SPACE.MIT.EDU"
    ELECTRONIC_MAIL_TYPE              = "INTERNET"
    PREFERENCE_ID                     = 1
  END_OBJECT                          = PERSONNEL_ELECTRONIC_MAIL

  OBJECT                              = PERSONNEL_ELECTRONIC_MAIL
    ELECTRONIC_MAIL_ID                = "PFORD"
    ELECTRONIC_MAIL_TYPE              = "NASAMAIL"
    PREFERENCE_ID                     = 2
  END_OBJECT                          = PERSONNEL_ELECTRONIC_MAIL

  OBJECT                              = PERSONNEL_ELECTRONIC_MAIL
    ELECTRONIC_MAIL_ID                = "JPLPDS::PFORD"
    ELECTRONIC_MAIL_TYPE              = "NSI/DECNET"
    PREFERENCE_ID                     = 3
 END_OBJECT                           = PERSONNEL_ELECTRONIC_MAIL

END_OBJECT                             = PERSONNEL
END
```

B.30 PERSONNEL_ELECTRONIC_MAIL

The PERSONNEL_ELECTRONIC_MAIL object, a sub-object of the PERSONNEL catalog object, provides electronic mail information for an individual. This object may be repeated if more than one electronic mail address is applicable.

B.30.1 Required Keywords

1. ELECTRONIC_MAIL_ID
2. ELECTRONIC_MAIL_TYPE
3. PREFERENCE_ID

B.30.2 Optional Keywords

None

B.30.3 Required Objects

None

B.30.4 Optional Objects

None

B.30.5 Example

See the example for the PERSONNEL object in Section B.28.5.

B.31 PERSONNEL_INFORMATION

The PERSONNEL_INFORMATION object, a sub-object of the PERSONNEL catalog object, provides name, address, telephone, and related information for an individual.

B.31.1 Required Keywords

1. ADDRESS_TEXT
2. ALTERNATE_TELEPHONE_NUMBER
3. FAX_NUMBER
4. FULL_NAME
5. INSTITUTION_NAME
6. LAST_NAME
7. NODE_ID
8. PDS_AFFILIATION
9. REGISTRATION_DATE
10. TELEPHONE_NUMBER

B.31.2 Optional Keywords

1. PDS_ADDRESS_BOOK_FLAG

B.31.3 Required Objects

None

B.31.4 Optional Objects

None

B.31.5 Example

See the example for the PERSONNEL object in Section B.28.5.

B.32 REFERENCE

The REFERENCE catalog object provides a citation and a unique identifier for every journal article, book, chapter, or other reference mentioned in a CATALOG object or one of its components (MISSION, INSTRUMENT HOST, INSTRUMENT, DATA SET, etc.).

One REFERENCE catalog object should be completed for each reference associated with a CATALOG (or component) object. Since the goal in generating REFERENCE catalog objects is to provide additional external long-term documentation, care should be exercised in selecting candidate references. Internal documents, informal memoranda, and other unpublished material should be avoided. A REFERENCE should cite published data, such as other PDS archives. Multiple REFERENCE catalog objects are often assembled into a single REF.CAT file to accompany an archive.

B.32.1 Required Keywords

1. REFERENCE_KEY_ID
2. REFERENCE_DESC

B.32.2 Optional Keywords

None

B.32.3 Required Objects

None

B.32.4 Optional Objects

None

B.32.5 Usage Notes

The following examples show how to cite various types of information in REFERENCE catalog objects for PDS archive products. PDS has elected to use the American Geophysical Union (AGU) reference citation formats. The information presented within this section was derived from and complies with AGU's formats for publication (see www.agu.org/pubs/references.html for more information). For assistance in determining the proper format for a citation that does not fit within one of the categories described here, contact a PDS Data Engineer.

B.32.5.1 Materials Appropriate for Inclusion in a REFERENCE Catalog Object

Each article, book, report, electronic collection (CD-ROM or electronic publication), thesis, or similar publication used in documenting a PDS archival product should be defined by a REFERENCE catalog object.

B.32.5.2 Materials Inappropriate for Inclusion in a REFERENCE Catalog Object

Unpublished materials such as personal communications, unpublished reports, papers presented at meetings, and manuscripts in preparation or submitted for publication but not yet formally accepted are not allowed in REFERENCE catalog objects; PDS does not consider them to be part of the literature. Likewise, internal memoranda and documents should be avoided unless they can be accessed by outside users. Papers accepted but without final publication data (volume, page numbers, dates, etc.) are discouraged since the information in the REFERENCE catalog object would be incomplete and need to be updated later.

In cases where it would be desirable to credit another author or group for contributions or prior work, an in-line text acknowledgement or citation is acceptable, even when the material is not readily accessible. If such material is required for understanding the archive, the normal constraints apply, however.

B.32.5.3 Reference List Citations

The most widely accessible source of a particular piece of material should be cited. For example, if an article exists as an internal publication and in a professional journal, the latter should be used in the REFERENCE catalog object.

B.32.5.4 Construction of REFERENCE_KEY_ID

1. For a single author, the REFERENCE_KEY_ID comprises the author's surname followed by the year of the publication (e.g., "SMITH1990").

2. For two authors, the REFERENCE_KEY_ID comprises the author's surname followed by "&" followed by the co-author's surname followed by the year of the publication (e.g., "LAUREL&HARDY1990").

3. For more than two authors, the REFERENCE_KEY_ID comprises the first author's surname followed by "ETAL" followed by the year of the publication (e.g., "SMITHETAL1990").

4. If the same author(s) published more than one paper in the same year, the following applies:

   a. The initial publication will have a REFERENCE_KEY_ID as formulated above (e.g., "SMITH1990"). Note that the reference uses an implicit "A".

b. Subsequent publications will use the next sequential letter to uniquely identify the reference:
   - the 2nd publication will be "SMITH1990B",
   - the 3rd publication will be "SMITH1990C".

5. The REFERENCE_KEY_ID value should be enclosed within double quotes.

6. For additional information on formulating a REFERENCE_KEY_ID, check the PDS Data Dictionary (http://pdsproto.jpl.nasa.gov/onlinecatalog/top.cfm).

B.32.5.5 Construction of REFERENCE_DESC

The information included in a REFERENCE catalog object will vary somewhat depending on the source. The following subsections describe the most common types; contact a PDS Data Engineer for assistance when encountering a type not described here. Details on constructing the components of a REFERENCE_DESC value are described in the next section.

B.32.5.5.1 Papers in Professional Journals and Other Articles

Citations of articles should include the following information in the order listed:

1. Name(s) of author or authors
2. Title of article
3. Name of periodical
4. Volume and/or issue number
5. First and last pages occupied by article
6. Year of publication

Example:

```
OBJECT                     = REFERENCE
 REFERENCE_KEY_ID          = "SCARF&GURNETT1977"
 REFERENCE_DESC            = "
    Scarf, F.L., and D.A. Gurnett, A plasma wave investigation for the
    Voyager mission, Space Sci. Rev., Vol. 21, No. 1, pp. 289-331, 1977."
 END_OBJECT                = REFERENCE
```

B.32.5.5.2 Books and Reports

Citations of books and reports should include the following information, in the order listed:

1. Name(s) of author or authors
2. Title of article or chapter (if only part of book or report is being cited)
3. Title of book or report
4. Volume number (if part of a multivolume series)
5. Edition (if not original)
6. Editor(s) (if any)

7. Report number(s)
8. Page numbers (if only part of book or report is being cited)
9. Publisher's name
10. City of publication
11. Year of publication

Examples:

```
OBJECT                          = REFERENCE
 REFERENCE_KEY_ID               = "FIELDETAL1989B"
 REFERENCE_DESC                 = "
    Field, S.W., S.E. Haggerty, and A.J. Erlank, Subcontinental
    Metasomatism in the Region of Jagersfontein, Springer-Verlag,
    New York, 1989."
END_OBJECT                      = REFERENCE

OBJECT                          = REFERENCE
 REFERENCE_KEY_ID               = "THOMPSON1985"
 REFERENCE_DESC                 = "
    Thompson, W.B., Preliminary investigation of the electrodynamic
    of a conducting tether, in Spacecraft Environmental Technology
    1983, edited by C. K. Purvis and C. P. Pike, NASA Conf. Publ.,
    Vol. 2359, pp. 649-662, National Aeronautics and Space Administration,
    Washington, DC, 1985."
END_OBJECT                      = REFERENCE
```

B.32.5.5.3 Electronic Publications

Certain types of electronic publications may be given as REFERENCE catalog objects. These include publications on electronic media such as CD-ROM and regularly issued, dated electronic journals. Data deposited at PDS and National Data Centers (e.g., NSSDC) may also be included. Because of the ephemeral nature of some electronic media, authors should consult a Data Engineer if the specific reference (e.g., a website) does not seem to have a traditional hardcopy analog.

B.32.5.5.3.1 Data Sets

REFERENCE catalog objects for data sets that are on deposit at PDS or National Data Centers (e.g., NSSDC) should include the following information, in the order listed:

1. Name of author or authors (e.g., Principal Investigator and/or Data Producer)
2. Name of the data set (e.g., DATA_SET_NAME)
3. Unique identifier of the data set (e.g., DATA_SET_ID)
4. Volume and/or issue number (e.g., VOLUME_SET_ID or VOLUME_ID) (optional)
5. Name of publisher or producer (e.g., NASA Planetary Data System)
6. Year of publication

Example:

```
OBJECT                      = REFERENCE
 REFERENCE_KEY_ID           = "LEVINETAL2000"
 REFERENCE_DESC             = "
    Levin, G.V., P.A. Straat, E.A. Guinness, P.G. Valko, J.H. King,
    and D.R. Williams, Viking Lander Labeled Release Data Archive,
    VL1/VL2-M-LCS-2-EDR-V1.0, USA_NASA_JPL_VL_9010, NASA
    Planetary Data System, 2000."
END_OBJECT                  = REFERENCE
```

B.32.5.5.3.2 Physical Media (CD-ROM / DVD-R)

REFERENCE catalog objects for physical media (e.g., CDs or DVDs) should include the following information, in the order listed:

1. Name of author or authors (e.g., Principal Investigator and/or Data Producer)
2. Name of the volume or volume set (e.g., VOLUME_NAME or VOLUME_SET_NAME)
3. Unique identifier of the volume or volume set (e.g., VOLUME_ID or VOLUME_SET_ID)
4. Name of publisher or producer (e.g., NASA Planetary Data System)
5. Year of publication

Example:

```
OBJECT                      = REFERENCE
 REFERENCE_KEY_ID           = "LEVINETAL2000B"
 REFERENCE_DESC             = "
Levin, G.V., P.A. Straat, E.A. Guinness, P.G. Valko, J.H. King,
and D.R. Williams, Viking Lander 1 Experiment Data Records (EDR)
Image Products, USA_NASA_JPL_VL_00xx, NASA Planetary Data
System, 2000."
END_OBJECT                  = REFERENCE
```

B.32.5.5.3.3 Electronic Journal Articles

Material published in regularly issued, dated electronic journals should be referenced similarly to printed papers (see Papers in Professional Journals and Other Articles, above). Because this aspect of the Internet is evolving rapidly, PDS does not offer specific recommendations; authors should contact a Data Engineer for current guidelines. Because the Internet is a dynamic environment and sites may change or move, PDS cautions that such electronic sources should have established a record of stability before being considered acceptable for use in REFERENCE catalog objects.

B.32.5.6 REFERENCE_DESC Components

B.32.5.6.1 Author Names

For the first author only, the surname is given first, followed by initials. Names of any co-authors appear in regular order: initials precede the co-author's surname. The word “and” precedes the last author’s name. Do not include white space between authors' initials (e.g., Kurth, W.S.) When the number of authors exceeds five, the author list may consist of the first five authors’ names and initials as usual, followed by “and N others”, where “N”, an arabic numeral, is the number of remaining authors.

Example:

```
OBJECT                          = REFERENCE
 REFERENCE_KEY_ID               = "KURTHETAL1982"
 REFERENCE_DESC                 = "
Kurth, W.S., F.L. Scarf, J.D. Sullivan, and D.A. Gurnett,
Detection of nonthermal continuum radiation in Saturn's
magnetosphere, Geophys. Res. Lett., Vol. 9, p. 889, 1982."
END_OBJECT                      = REFERENCE
```

B.32.5.6.2 Journal Titles

PDS uses the same guidelines as the AGU which were established by the ***Chemical Abstracts Service Source Index*** in abbreviating the names of serial publications and reports. One word titles (e.g., Science, Icarus, Nature) are not abbreviated. Articles, conjunctions, prepositions, hyphens, parentheses, commas, and accents are omitted in abbreviated titles. Apostrophes in transliterated titles are retained.

Examples of common journal titles in planetary science include:

- Adv. Space Res.
- Geophys. Res. Lett.
- J. Geophys. Res.
- Rev. Geophys.
- Radio Sci.
- Space Sci. Rev.

Other examples include:

- AAPG Bull.
- Anal. Chem.
- Ann. Geophys.
- Ann. Glaciol.
- Appl. Opt.
- Appl. Spectrosc.
- Astrophys. J.
- Bull. Int. Assoc. Eng. Geol.
- Bull. Mar. Sci.

- Can. J. Phys.
- Chem. Geol.
- Contrib. Mineral. Petrol.
- Earth Planet. Sci. Lett.
- Geochim. Cosmochim. Acta
- Geol. Soc. Am. Bull.
- IEEE Trans. Geosci. Remote Sens.
- IEEE Trans. Nucl. Sci.
- Int. J. Rock Mech. Min. Sci. Geomech. Abstr.
- J. Atmos. Chem.
- J. Atmos. Oceanic Technol.
- J. Atmos. Sci.
- J. Atmos. Terr. Phys.
- J. Fluid Mech.
- J. Geomagn. Geoelectr.
- J. High Resolut. Chromatogr.
- J. Petrol.
- J. Phys. Oceanogr.
- Mon. Weather Rev.
- Phys. Fluids
- Philos. Trans. R. Soc. London Ser. A
- Planet. Space Sci.
- Q. J. R. Meteorol. Soc.
- Remote Sens. Environ.
- Science

B.33 SOFTWARE

The SOFTWARE catalog object provides general information about a software tool including description, availability information, and dependencies.

The SOFTWARE catalog object is completed for each software program registered in the PDS Software Inventory. This Inventory includes software available within the planetary science community, including software on PDS archive volumes. Of interest are any applications, tools, or libraries that have proven useful for the display, analysis, formatting, transformation, or preparation of either science data or meta-data for the PDS archives.

B.33.1 Required Keywords

1. SOFTWARE_ID
2. SOFTWARE_VERSION_ID

B.33.2 Optional Keywords

None

B.33.3 Required Objects

1. SOFTWARE_INFORMATION
2. SOFTWARE_ONLINE
3. SOFTWARE_PURPOSE

B.33.4 Optional Objects

None

B.33.5 Example

```
/* Template: Software Template                      Rev: 1998-12-01              */

/* Note: This template should be completed to register software in the          */
/*       PDS Software Inventory.                                                 */

PDS_VERSION_ID                       = PDS3
LABEL_REVISION_NOTE                  = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                          = STREAM

OBJECT                               = SOFTWARE
  SOFTWARE_ID                        = NASAVIEW
```

```
  SOFTWARE_VERSION_ID                 = "V1R2B"

  OBJECT                              = SOFTWARE_INFORMATION
    SOFTWARE_NAME                     = "NASAVIEW - PDS DATA PRODUCT ACCESS TOOL
                                         V1.2B"
    DATA_FORMAT                       = PDS
    SOFTWARE_LICENSE_TYPE             = PUBLIC_DOMAIN
    TECHNICAL_SUPPORT_TYPE            = FULL
    REQUIRED_STORAGE_BYTES            = "1.8MB"
    PDS_USER_ID                       = SHUGHES
    NODE_ID                           = CN
    SOFTWARE_DESC                     = "


  Software Overview
  =================
    NasaView Version 1.2b is a PDS Image display program developed for the following platforms:
    (a) PC / Win32
    (b) Unix / Sun OS

    NasaView is capable of accessing and displaying all images, tables, cubes, and histograms in
    the PDS archive. This release has been tested using Galileo, Magellan, Viking, MDIM, Voyager,
    IHW LSPN, and Clementine uncompressed images.

    NasaView is planned as a PDS data product object display utility that will run on SUN, MAC,
    and PC platforms in a GUI environment.

    This application was built using the Label Library Light (L3), Object Access Library (OAL),
    and the XVT Development Solution for C package. Label Library Light parses PDS ODL labels and
    creates an in-memory representation of the label information. The Object Access Library uses
    the parse-tree and accesses the actual PDS object. The XVT Development Solution supplies the
    cross platform GUI and an Object-oriented environment. XVT allows the definition of visual
    objects such as Windows and Menus and associates events and code with them.


  Available Support Material
  ==========================
       BINARIES


  Programming Language
  ====================
       SUN_C


  Platforms Supported
  ===================
       PC / Microsoft Win95, Win98, NT4.0


  Support Software Required / Used
  ================================
       X_WINDOWS

  END_OBJECT                          = SOFTWARE_INFORMATION

  OBJECT                              = SOFTWARE_ONLINE
    ON_LINE_IDENTIFICATION            = "http://pds.jpl.nasa.gov/license.html"
    ON_LINE_NAME                      = "NASAVIEW REVISION 2 BETA"
    NODE_ID                           = CN
    PROTOCOL_TYPE                     = URL
    PLATFORM                          = "PC/WIN32"
  END_OBJECT                          = SOFTWARE_ONLINE

  OBJECT                              = SOFTWARE_PURPOSE
    SOFTWARE_PURPOSE                  = DISPLAY
  END_OBJECT                          = SOFTWARE_PURPOSE

END_OBJECT                            = SOFTWARE
END
```

B.34 SOFTWARE_INFORMATION

The SOFTWARE_INFORMATION object, a sub-object of SOFTWARE catalog object, provides basic identification and operating system information associated with a specific SOFTWARE object.

B.34.1 Required Keywords

1. DATA_FORMAT
2. NODE_ID
3. PDS_USER_ID
4. REQUIRED_STORAGE_BYTES
5. SOFTWARE_DESC
6. SOFTWARE_LICENSE_TYPE
7. SOFTWARE_NAME
8. TECHNICAL_SUPPORT_TYPE

B.34.2 Optional Keywords

None

B.34.3 Required Objects

None

B.34.4 Optional Objects

None

B.34.5 Example

See the example for the SOFTWARE object in Section B.32.5.

B.35 SOFTWARE_ONLINE

The SOFTWARE_ONLINE object, a sub-object of SOFTWARE catalog object, provides identifying information for each PDS node providing access to a particular SOFTWARE object.

B.35.1 Required Keywords

1. NODE_ID
2. ON_LINE_IDENTIFICATION
3. ON_LINE_NAME
4. PLATFORM
5. PROTOCOL_TYPE

B.35.2 Optional Keywords

None

B.35.3 Required Objects

None

B.35.4 Optional Objects

None

B.35.5 Example

See the example for the SOFTWARE object in Section B.32.5.

B.36 SOFTWARE_PURPOSE

The SOFTWARE_PURPOSE object, a sub-object of SOFTWARE catalog object, describes the functionality provided by a specific SOFTWARE object.

B.36.1 Required Keywords

1. SOFTWARE_PURPOSE

B.36.2 Optional Keywords

None

B.36.3 Required Objects

None

B.36.4 Optional Objects

None

B.36.5 Example

See the example for the SOFTWARE object in Section B.32.5.

B.37 TARGET

The TARGET catalog object provides basic descriptive information for a single observational target.

B.37.1 Required Keywords

1. TARGET_NAME

B.37.2 Optional Keywords

None

B.37.3 Required Objects

1. TARGET_INFORMATION

B.37.4 Optional Objects

1. TARGET_REFERENCE_INFORMATION

B.37.5 Usage Notes

One TARGET_INFORMATION catalog object must be completed for each target. A TARGET_REFERENCE_INFORMATION catalog object is required for each individual reference associated with the target. All references should be included that are relevant to providing more detailed / specific target information; such as, type of target, orbit direction, description of the target, etc. These references may include published articles, books, papers, electronic publications, etc.

B.37.6 Example

```
/* Template: Target Template                     Rev: 1995-01-01    */

/* Note:  The following template is used for the                    */
/*        submission of a target to the PDS                         */

PDS_VERSION_ID                       = PDS3
LABEL_REVISION_NOTE                  = "1998-07-01, Richard Simpson (STANFORD), initial;"
RECORD_TYPE                          = STREAM
```

```
OBJECT                                = TARGET
  TARGET_NAME                         = JUPITER

  OBJECT                              = TARGET_INFORMATION
    TARGET_TYPE                       = PLANET
    PRIMARY_BODY_NAME                 = SUN
    ORBIT_DIRECTION                   = PROGRADE
    ROTATION_DIRECTION                = PROGRADE
    TARGET_DESC                       = "

  A_AXIS_RADIUS               : 71492.000000
  B_AXIS_RADIUS               : 71492.000000
  BOND_ALBEDO                 : UNK
  C_AXIS_RADIUS               : 66854.000000
  FLATTENING                  : 0.006500
  MAGNETIC_MOMENT             : 155000000000000000000.000000
  MASS                        : 1898799999999999953652202602496.000000
  MASS_DENSITY                : 1.330000
  MINIMUM_SURFACE_TEMPERATURE : UNK
  MAXIMUM_SURFACE_TEMPERATURE : UNK
  MEAN_SURFACE_TEMPERATURE    : UNK
  EQUATORIAL_RADIUS           : 71492.000000
  MEAN_RADIUS                 : 69911.000000
  SURFACE_GRAVITY             : 25.900000
  REVOLUTION_PERIOD           : 4333.000000
  POLE_RIGHT_ASCENSION        : 268.000000
  POLE_DECLINATION            : 64.500000
  SIDEREAL_ROTATION_PERIOD    : 0.410000
  MEAN_SOLAR_DAY              : 0.410000
  OBLIQUITY                   : 3.100000
  ORBITAL_ECCENTRICITY        : 0.048000
  ORBITAL_INCLINATION         : 1.300000
  ORBITAL_SEMIMAJOR_AXIS      : 778376719.000000
  ASCENDING_NODE_LONGITUDE    : 100.500000
  PERIAPSIS_ARGUMENT_ANGLE    : 275.200000"

  END_OBJECT                          = TARGET_INFORMATION

  OBJECT                              = TARGET_REFERENCE_INFORMATION
    REFERENCE_KEY_ID                  = "XYZ95"
  END_OBJECT                          = TARGET_REFERENCE_INFORMATION

END_OBJECT                            = TARGET
END
```

B.38 TARGET_INFORMATION

The TARGET_INFORMATION object, a sub-object of the TARGET catalog object, provides physical and dynamic parameters of the target.

B.38.1 Required Keywords

1. ORBIT_DIRECTION
2. PRIMARY_BODY_NAME
3. ROTATION_DIRECTION
4. TARGET_DESC
5. TARGET_TYPE

B.38.2 Optional Keywords

None

B.38.3 Required Objects

None

B.38.4 Optional Objects

None

B.38.5 Example

See the example for the TARGET object in Section B.36.5.

B.39 TARGET_REFERENCE_INFORMATION

The TARGET_REFERENCE_INFORMATION object, a sub-object of the TARGET catalog object, associates a reference with a target. A separate object must be completed for each reference cited within the TARGET catalog object.

A separate REFERENCE catalog object is completed to provide the associated citation for each reference.

B.39.1 Required Keywords

1. REFERENCE_KEY_ID

Note: If there are no relevant references to cite, the REFERENCE_KEY_ID should have a value of "N/A".

B.39.2 Optional Keywords

None

B.39.3 Required Objects

None

B.39.4 Optional Objects

None

B.39.5 Usage Notes

NOTE: The following are recommended as the minimum set of information needed to describe a target adequately. Additional information may be added as desired. If any of the information not be available or is not known then, consult Chapter 17, Usage of N/A, UNK, and NULL.

```
A_AXIS_RADIUS
B_AXIS_RADIUS
BOND_ALBEDO
C_AXIS_RADIUS
FLATTENING
MAGNETIC_MOMENT
```

```
MASS
MASS_DENSITY
MINIMUM_SURFACE_TEMPERATURE
MAXIMUM_SURFACE_TEMPERATURE
MEAN_SURFACE_TEMPERATURE
EQUATORIAL_RADIUS
MEAN_RADIUS
SURFACE_GRAVITY
REVOLUTION_PERIOD
POLE_RIGHT_ASCENSION
POLE_DECLINATION
SIDEREAL_ROTATION_PERIOD
MEAN_SOLAR_DAY
OBLIQUITY
ORBITAL_ECCENTRICITY
ORBITAL_INCLINATION
ORBITAL_SEMIMAJOR_AXIS
ASCENDING_NODE_LONGITUDE
PERIAPSIS_ARGUMENT_ANGLE
```

B.39.6 Example

See the example for the TARGET object in Section B.36.5.

Appendix C. Internal Representation of Data Types

This appendix contains the detailed internal representations of the PDS standard data types listed in Table 3.2 of the *Data Type Definitions* chapter of this document.

Chapter Contents

Appendix C. Internal Representation of Data Types C-1

C.1 MSB_INTEGER C-2
C.2 MSB_UNSIGNED_INTEGER C-4
C.3 LSB_INTEGER C-6
C.4 LSB_UNSIGNED_INTEGER C-8
C.5 IEEE_REAL C-10
C.6 IEEE_COMPLEX C-13
C.7 PC_REAL C-14
C.8 PC_COMPLEX C-17
C.9 VAX_REAL, VAXG_REAL C-18
C.10 VAX_COMPLEX, VAXG_COMPLEX C-22
C.11 MSB_BIT_STRING C-23
C.12 LSB_BIT_STRING C-25

C.1 MSB_INTEGER

Aliases: INTEGER
MAC_INTEGER
SUN_INTEGER

This section describes the signed integers stored in Most Significant Byte first (MSB) order. In this section the following definitions apply:

b0 – b3 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3)

i-sign Integer sign bit (bit 7 in the highest-order byte

i0 – i3 Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7) in the following way:

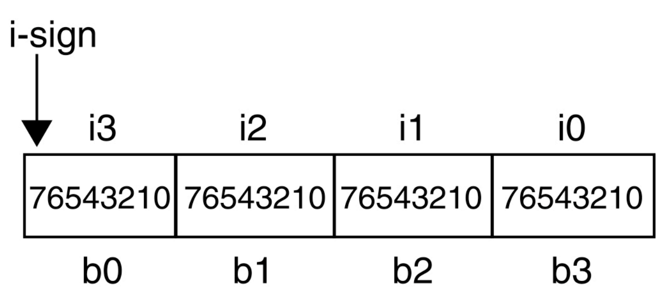
4-byte integers:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-6 represent 2**24 through 2**30

2-byte integers:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-6 represent 2**8 through 2**14

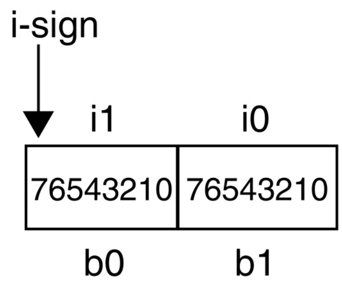
1-byte integers:
In i0, bits 0-6 represent 2**0 through 2**6

Negative integers are represented in two's complement.

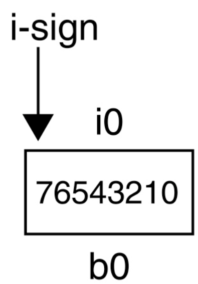
C.1.1 MSB 4-byte Integer



C.1.2 MSB 2-byte Integer



C.1.3 MSB 1-byte Integer



C.2 MSB_UNSIGNED_INTEGER

Aliases: UNSIGNED_INTEGER
MAC_UNSIGNED_INTEGER
SUN_UNSIGNED_INTEGER

This section describes unsigned integers stored in Most Significant Byte first (MSB) format. In this section the following definitions apply:

b0 – b3 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

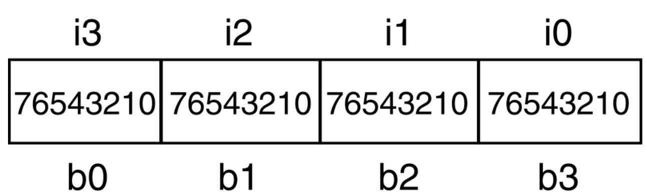
i0 – i3 Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-7 represent 2**24 through 2**31

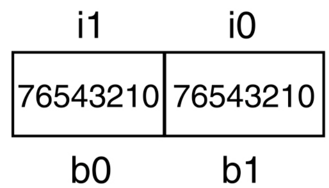
2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15

1-byte:
In i0, bits 0-7 represent 2**0 through 2**7

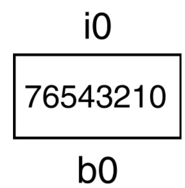
C.2.1 MSB 4-byte Unsigned Integers



C.2.2 MSB 2-byte Unsigned Integers



C.2.3 MSB 1-byte Unsigned Integers



C.3 LSB_INTEGER

Aliases: PC_INTEGER
VAX_INTEGER

This section describes signed integers stored in Least Significant Byte first (LSB) order. In this section the following definitions apply:

b0 – b3 Arrangement of bytes as they appear when reading a file (e.g., read byte b0 first, then b1, b2 and b3)

i-sign Integer sign bit – bit 7 in the highest order byte

i0 – i3 Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

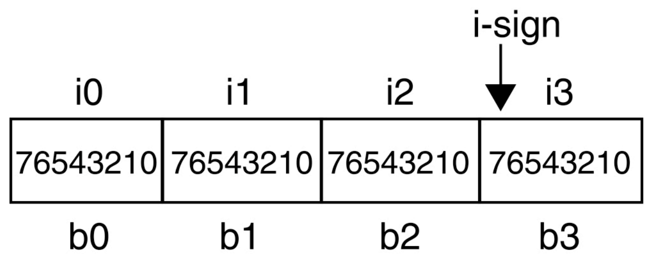
4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-6 represent 2**24 through 2**30

2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-6 represent 2**8 through 2**14

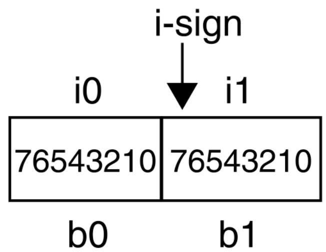
1-byte:
In i0, bits 0-6 represent 2**0 through 2**6

All negative values are represented in two's complement.

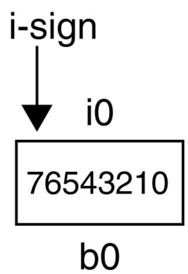
C.3.1 LSB 4-byte Integers



C.3.2 LSB 2-byte Integers



C.3.3 LSB 1-byte Integers



C.4 LSB_UNSIGNED_INTEGER

Aliases: PC_UNSIGNED_INTEGER
VAX_UNSIGNED_INTEGER

This section describes unsigned integers stored in Least Significant Byte first (LSB) format. In this section the following definitions apply:

b0 – b3 Arrangement of bytes as they appear when reading a file (e.g., read byte b0 first, then b1, b2 and b3)

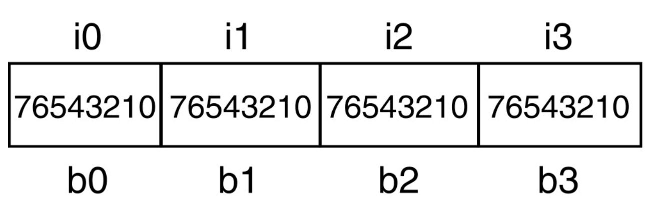
i0 – i3 Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., lowest value = bit 0, highest value = bit 7), in the following way:

4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-7 represent 2**24 through 2**31

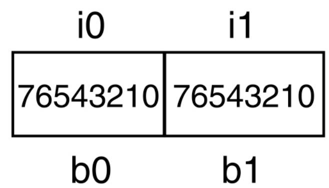
2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15

1-byte:
In i0, bits 0-7 represent 2**0 through 2**7

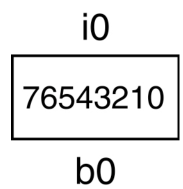
C.4.1 LSB 4-byte Unsigned Integers



C.4.2 LSB 2-byte Unsigned Integers



C.4.3 LSB 1-byte Unsigned Integers



C.5 IEEE_REAL

Aliases: FLOAT
REAL
MAC_REAL
SUN_REAL

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

b0 – b9 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, b3, etc.)

m-sign Mantissa sign bit

int-bit In 10-byte real format only, the integer part of the mantissa, assumed to be "1" in other formats, is explicitly indicated by this bit

e0 – e1 Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., 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:

10-bytes (temporary):
In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**8 through 2**14

Exponent bias = 16383

8-bytes (double precision):
In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1023

4-bytes (single precision):
In e0, bit 7 represent 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 127

m0 – m7 Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

10-bytes (temporary):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55
In m7, bits 7-0 represent 1/2**56 through 1/2**63

8-bytes (double precision):
In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

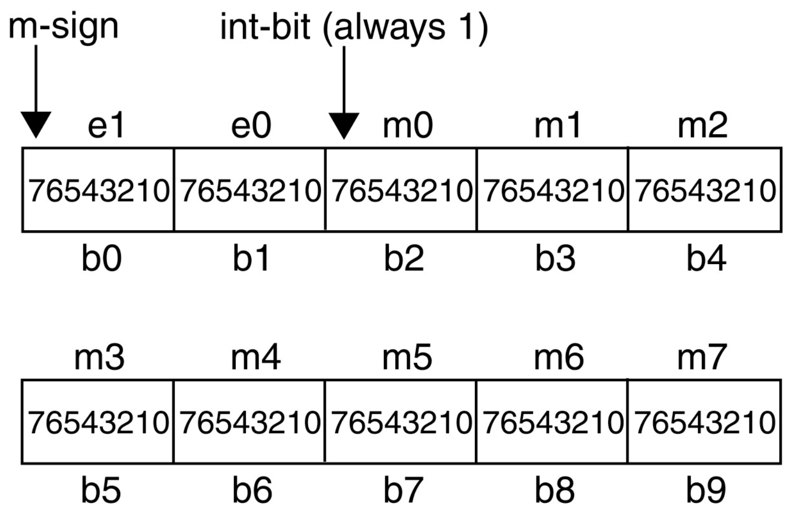
4-bytes (single precision):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

The following representations all follow this format:

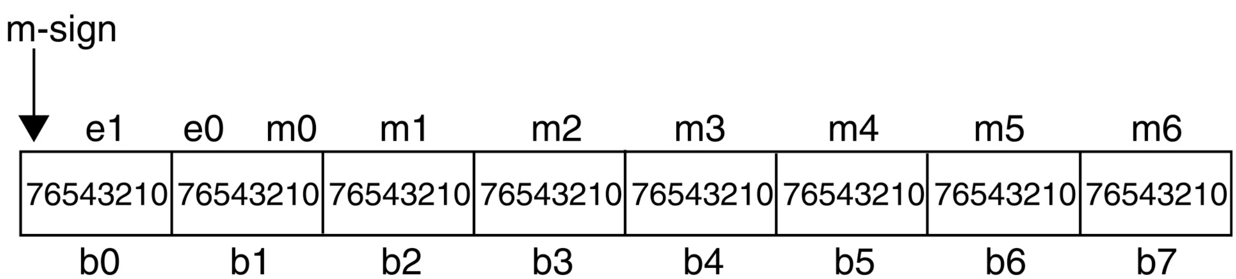
1.*mantissa* × 2**(*exponent – bias)*

Note that the integer part ("1.") is implicit in all formats except the 10-byte (temporary) real format, 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).

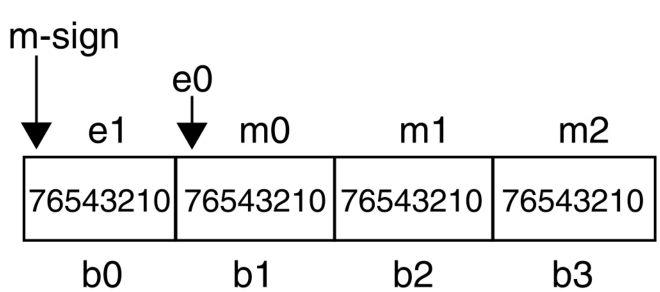
C.5.1 IEEE 10-byte (Temporary) Real Numbers



C.5.2 IEEE 8-byte (Double Precision) Real Numbers



C.5.3 IEEE 4-byte (Single Precision) Real Numbers



C.6 IEEE_COMPLEX

Aliases: COMPLEX
MAC_COMPLEX
SUN_COMPLEX

IEEE complex numbers consist of two IEEE_REAL format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using IEEE_REAL formats, see Section C.5.

C.7 PC_REAL

Aliases: None

This section describes the internal storage format corresponding to the PC_REAL data type. In this section the following definitions apply:

b0 – b9 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

m-sign Mantissa sign bit

int-bit In 10-byte real format only, the integer part of the mantissa, assumed to be “1” in other formats, is explicitly indicated by this bit.

e0 – e1 Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., 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:

10-bytes (temporary):
In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**8 through 2**14

Exponent bias = 16383

8-bytes (double precision):
In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1023

4-bytes (single precision):
In e0, bit 7 represent 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 127

m0 – m7 Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

10-bytes (temporary):

In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55
In m7, bits 7-0 represent 1/2**56 through 1/2**63

8-bytes (double precision):
In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

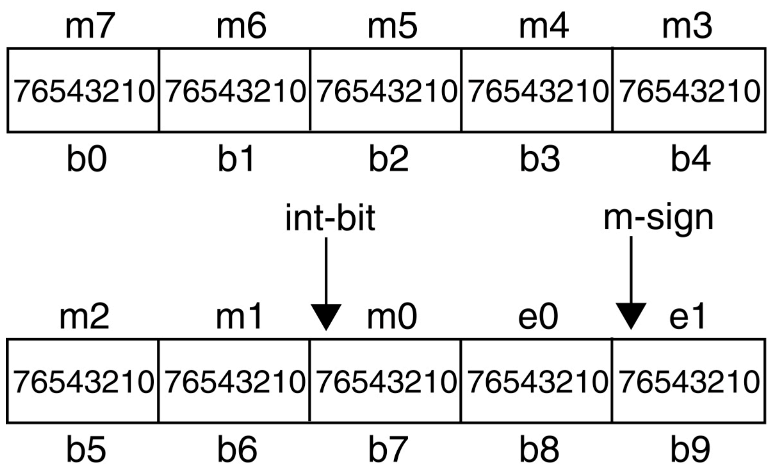
4-bytes (single precision):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

The following representations all follow this format:

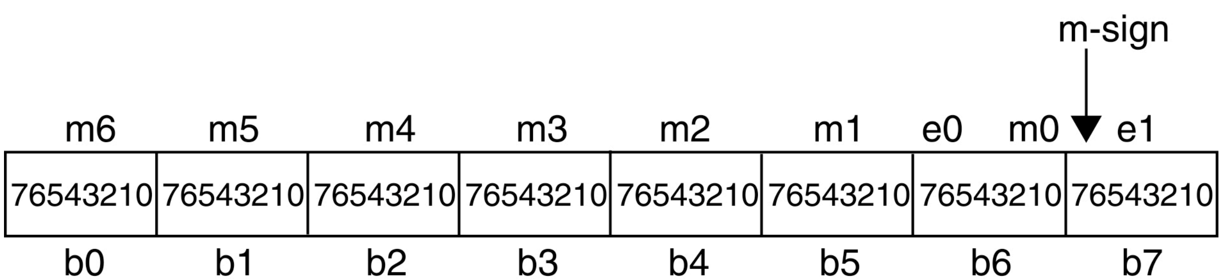
$$1.\mathit{mantissa} \times 2^{**}(\mathit{exponent} - \mathit{bias})$$

Note that the integer part ("1.") is implicit in all formats except the 10-byte (temporary) real format, 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).

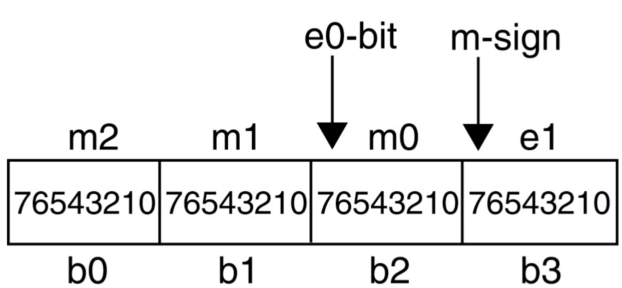
C.7.1 PC 10-byte (Temporary) Real Numbers



C.7.2 PC 8-byte (Double Precision) Real Numbers



C.7.3 PC 4-byte (Single Precision) Real Numbers



C.8 PC_COMPLEX

Aliases: None

PC complex numbers consist of two PC_REAL format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using PC_REAL formats, see Section C.7.

C.9 VAX_REAL, VAXG_REAL

Aliases: VAX_DOUBLE for VAX_REAL only.
No aliases for VAXG_REAL

This section describes the internal format corresponding to the VAX_REAL and VAXG_REAL data types. In this section the following definitions apply:

b0 – b15 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

m-sign Mantissa sign bit

e0 – e1 Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left (e.g., 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:

16-bytes (H-type, quad precision):
In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**8 through 2**14

Exponent bias = 16385

8-bytes (G-type, double precision):
In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1025

8-bytes (D-type, double precision):
In e0, bit 7 represents 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 129

4-bytes (F-type, single precision):
In e0, bit 7 represent 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 129

m0 – m13 Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fraction. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

16-bytes (H-type, quad precision):
In m0, bits 7-0 represent 1/2**1 through 1/2**8
In m1, bits 7-0 represent 1/2**9 through 1/2**16
In m2, bits 7-0 represent 1/2**17 through 1/2**24
In m3, bits 7-0 represent 1/2**25 through 1/2**32
In m4, bits 7-0 represent 1/2**33 through 1/2**40
In m5, bits 7-0 represent 1/2**41 through 1/2**48
In m6, bits 7-0 represent 1/2**49 through 1/2**56
In m7, bits 7-0 represent 1/2**57 through 1/2**64
In m8, bits 7-0 represent 1/2**65 through 1/2**72
In m9, bits 7-0 represent 1/2**73 through 1/2**80
In m10, bits 7-0 represent 1/2**81 through 1/2**88
In m11, bits 7-0 represent 1/2**89 through 1/2**96
In m12, bits 7-0 represent 1/2**97 through 1/2**104
In m13, bits 7-0 represent 1/2**105 through 1/2**112

8-bytes (G-type, double precision):
In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

8-bytes (D-type, double precision):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55

4-bytes (F-type, single precision):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

The following representations all follow this format:

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

Note that the integer part ("1.") is implicit in all formats except the 10-byte (temporary) real format, 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).

C.9.1 VAX 16-byte H-type (Quad Precision) Real Numbers

m-sign

e0	e1	m1	m0	m3	m2	m5	m4
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

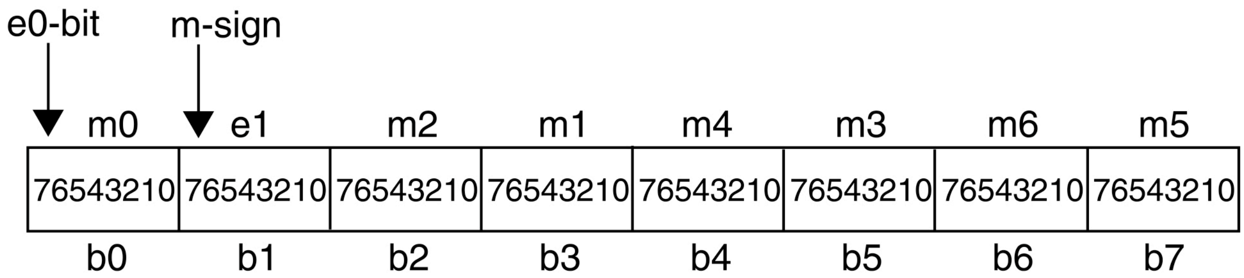
m7	m6	m9	m8	m11	m10	m13	m12
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b8	b9	b10	b11	b12	b13	b14	b15

C.9.2 VAX 8-byte G-type (Double Precision) Real Numbers

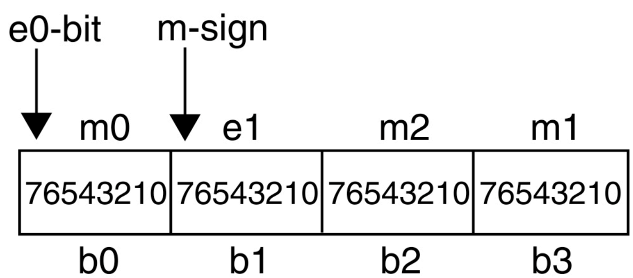
m-sign

e0 m0	e1	m2	m1	m4	m3	m6	m5
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

C.9.3 VAX 8-byte D-type (Double Precision) Real Numbers



C.9.4 VAX 4-byte F-type (Single Precision) Real Numbers



C.10 VAX_COMPLEX, VAXG_COMPLEX

Aliases: None

VAX complex numbers consist of two VAX_REAL (or VAXG_REAL) format numbers of the same precision, contiguous in memory. The first number represents the real part and the second the imaginary part of the complex value.

For more information on using VAX_REAL formats, see Section C.9.

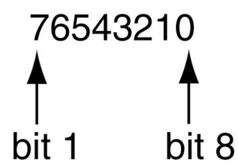
C.11 MSB_BIT_STRING

Aliases: None

This section describes the storage format for bit strings stored in Most Significant Byte first (MSB) format. In this section the following definitions apply:

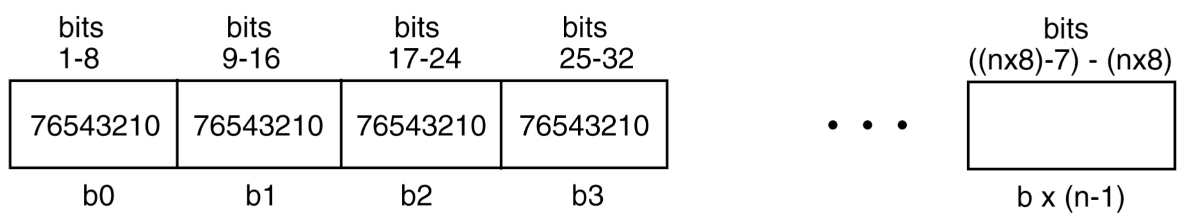
b0 – b3	Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

The bits within a byte are numbered from left to right, as shown below:

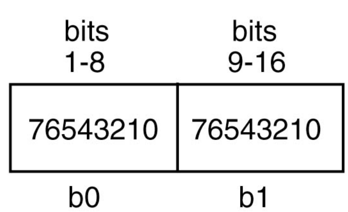


Note that in the case of MSB bit strings, no byte-swapping is required. That is, the physical storage order of the bytes is identical to the logical order.

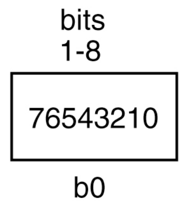
C.11.1 MSB *n*-byte Bit Strings



C.11.2 MSB 2-byte Bit String



C.11.3 MSB 1-byte Bit String



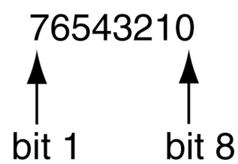
C.12 LSB_BIT_STRING

Aliases: VAX_BIT_STRING

This section describes the structure of bit strings stored in Least Significant Byte first (LSB) order. In this section, the following definitions apply:

b0 – b3 Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2 and b3)

The bits within a byte are numbered from left to right, as shown below:



Note that for LSB bit strings byte-swapping is required to convert the storage order of bytes to the logical order.

C.12.1 LSB 4-byte Bit String

Physical order (as read from the file):

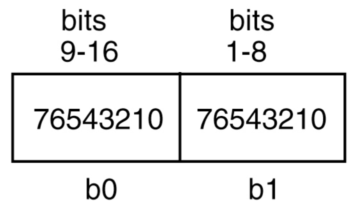
bits 25-32	bits 17-24	bits 9-16	bits 1-8
76543210	76543210	76543210	76543210
b0	b1	b2	b3

Logical order (after byte-swapping):

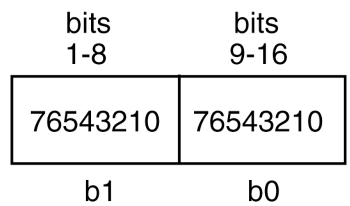
bits 1-8	bits 9-16	bits 17-24	bits 25-32
76543210	76543210	76543210	76543210
b3	b2	b1	b0

C.12.2 LSB 2-byte Bit String

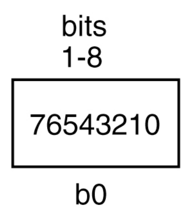
Physical order (as read from the file):



Logical order (after byte-swapping):



C.12.3 LSB 1-byte Bit String



Note that in this degenerate case no byte-swapping is required.

Appendix D. Examples of Required Files

The examples in this Appendix are based on existing or planned PDS archive volumes, and have been modified to reflect the most recent version of the PDS standards.

Chapter Contents

Appendix D. Examples of Required Files D-1

D.1 AAREADME.TXT D-2
D.2 INDXINFO.TXT D-8
D.3 SOFTINFO.TXT D-9
D.4 VOLDESC.CAT D-13

D.1 AAREADME.TXT

Each PDS archive volume shall include an "AAREADME.TXT" file that contains an overview of the contents and structure of the volume. An annotated outline is provided here as guidance for compiling this file.

D.1.1 Annotated Outline

I. PDS TEXT Object (must appear in an attached or detached label)

II. Volume Title

III. Contents

1. Introduction
   a. Science data content
   b. Conformance to PDS standards
   c. Document or institutional references for additional science information

2. Volume format
   a. Computer systems that can access the volume
   b. International standards to which the volume conforms

3. File formats
   a. Data record formats
   b. Specifications for specialized files (e.g., Postscript)
   c. Description of PDS objects, pointers, etc.

4. Volume contents
   a. Directory structure of the volume

5. Recommended CD-ROM drives (if applicable)
   a. Driver descriptions and notes for all appropriate computer platforms

6. Errata (if applicable)
   a. Known errors, cautionary notes, disclaimers, etc.
   b. Reference to the ERRATA.TXT file on the volume or online

7. Contacts
   a. Names and addresses of people or organizations to contact for questions concerning science data, technical support, data product generation and labelling, etc.

D.1.2 Example

The following is an example of an AAREADME.TXT file used on a PDS archive volume that does not use the logical volume construct. Note that section 3 in the example would need to be updated if logical volumes were present.

```
PDS_VERSION_ID                              = PDS3

RECORD_TYPE                                 = FIXED_LENGTH
RECORD_BYTES                                = 80
SPACECRAFT_NAME                             = MAGELLAN
TARGET_NAME                                 = VENUS
OBJECT                                      = TEXT
  PUBLICATION_DATE                          = 1994-06-01
  NOTE                                      = "MAGELLAN LOSAPDR ARCHIVE CD-WO"
END_OBJECT                                  = TEXT
END


MAGELLAN LOSAPDR ARCHIVE CD-WO

1. Introduction

This CD-WO contains Magellan Cycle 4 LOSAPDR (Line of Sight Acceleration Profile Data Record)
products. It also contains documentation which describe the LOSAPDRs. Each LOSAPDR product
contains the results from processing of radio tracking data of the Magellan spacecraft. There
are 866 LOSAPDRs on this volume.

The LOSAPDR products archived on this volume are the exact products released by the Magellan
Project. Supporting documentation and label files conform to the Planetary Data System (PDS)
Standards, Version 3.0, Jet Propulsion Laboratory (JPL) document JPL D-7669.

Additional information about the Magellan gravity experiment, including the acquisition,
processing, and quality of the LOSAPDR data, can be found in JPL documents that are available
from the PDS Geosciences Node, Washington University, St. Louis, MO.


2. Disk Format

The disk has been formatted so that a variety of computer systems (e.g. IBM PC, Macintosh,
Sun) may access the data. Specifically, it is formatted according to the ISO 9660 level 1
Interchange Standard. For further information, refer to the ISO 9660 Standard Document: RF#
ISO 9660-1988, 15 April 1988.


3. File Formats

Each orbit for which gravity data exists is represented by one LOSAPDR data file. The LOSAPDR
is an ASCII file. The data file contains 3 tables: 1) HEADER_TABLE; 2) TIMES_TABLE; and 3)
RESULTS_TABLE. The HEADER_TABLE is a single-row multi-column table containing information on
initial values, control parameters, and simple calculations required by the program that
generates the data files. The TIMES_TABLE is a single column containing exact times bounding
spline intervals to the Doppler residuals. The number of rows is variable. The RESULTS_TABLE
contains the results from spline fits to Doppler residuals. Each row in the table contains
times, Doppler residuals, spacecraft position and velocity information, and inferred
spacecraft acceleration. The data files are described by PDS labels embedded at the beginning
of the file. Further information on LOSAPDR file formats and contents can also be obtained
from the Magellan Software Interface Specification (SIS) document NAV-138. A copy of the
document is stored on this disk as file LOSAPDR.TXT in the DOCUMENT directory.

All document files and detached label files contain 80-byte fixed-length records, with a
carriage return character (ASCII 13) in the 79th byte and a line feed character (ASCII 10) in
the 80th byte. This allows the files to be read by the MacOS, DOS, Unix, and VMS operating
systems. All tabular files are also described by PDS labels, either embedded at the beginning
of the file or detached. If detached, the PDS label file has the same name as the data file it
describes, with the extension .LBL; for example, the file INDEX.TAB is accompanied by the
detached label file INDEX.LBL in the same directory.

Tabular files are formatted so that they may be read directly into many database management
systems on various computers. All fields are separated by commas, and character fields are
enclosed in double quotation marks ("). Character fields are left justified, and numeric
fields are right justified. The "start byte" and "bytes" values listed in the labels do not
include the commas between fields or the quotation marks surrounding character fields. The
```

records are of fixed length, and the last two bytes of each record contain the ASCII carriage return and line feed characters. This allows a table to be treated as a fixed length record file on computers that support this file type and as a normal text file on other computers.

A PostScript file, REPORT.PS, is included on this volume. This PostScript document is a validation report that lists all LOSAPDRs, and gives specific information, comments, and the status of each data file after a quality check and validation at the PDS Geophysics Subnode. The document is described by the detached label file, REPORT.LBL. The document can also be viewed by a Display PostScript program and can be printed out from a PostScript printer. The ASCII text version of the PostScript file is REPORT.ASC.

PDS labels are object-oriented. The object to which the label refers (e.g., IMAGE, TABLE, etc.) is denoted by a statement of the form:

```
^object = location
```

in which the carat character (^, also called a pointer in this context) indicates that the object starts at the given location. In an attached label, the location is an integer representing the starting record number of the object (the first record in the file is record 1). In a detached label, the location denotes the name of the file containing the object, along with the starting record or byte number. For example:

```
^TABLE = "INDEX.TAB"
```

indicates that the TABLE object points to the file INDEX.TAB .

Pointers to data objects are always required to be located in the same directory as the label file, so the file INDEX.TAB in this example is located in the same directory as the detached label file.

Other types of pointer statements can also be found on this volume. To resolve the pointer statement, first look in the same directory as the file containing the pointer statement. If the pointer is still unresolved, look in the following top level directory:
^ STRUCTURE - LABEL directory
^ CATALOG - CATALOG directory
^DATA_SET_MAP_PROJECTION - CATALOG directory
^DESCRIPTION - DOCUMENT directory.

Below is a list of the possible formats for the ^object keyword.

```
^object = n
^object = n<BYTES>
^object = "filename.ext"
^object = ("filename.ext",n)
^object = ("filename.ext",n<BYTES>)
```

where

```
n          is the starting record or byte number of the object,
           counting from the beginning of the file (record 1,
           byte 1)
<BYTES>    indicates that the number given is in units of bytes
filename   is the upper-case file name
ext        is the upper-case file extension
```

4. CD-ROM Contents

The files on this CD-ROM are organized in one top-level directory with several subdirectories. The following table shows the structure and content of these directories. In the table, directory names are enclosed in square brackets ([]), upper-case letters indicate an actual directory or file name, and lower-case letters indicate the general form of a set of directory or file names.

```
FILE                               CONTENTS

Top-level directory
|
|- AAREADME.TXT           The file you are reading.
|
|- ERRATA.TXT             Description of known anomalies and errors
|                         present on this volume.
|
|- VOLDESC.CAT            A description of the contents of this CD-
```

```
|                              ROM volume in a format readable
|                              by both humans and computers.
|
|- [CATALOG]                   A directory containing information about the
|      |                       LOSAPDR dataset.
|      |
|      |- CATALOG.CAT          PDS catalog objects. Mission, spacecraft
|      |                       and instrument descriptions.
|      |
|      |- CATINFO.TXT          Description of files in the CATALOG
|      |                       directory.
|      |
|      |- DATASET.CAT          PDS dataset catalog object. A description
|                              of the dataset, parameters, processing, data
|                              coverage and quality.
|
|- [DATA]                      A directory containing LOSAPDR data files.
|      |- [mmmmnnnn]           Directories containing LOSAPDR data files
|      |                       for orbits between 'mmmm' and 'nnnn'.
|      |
|      |     |- L0mmmm.001     LOSAPDR file for orbit number 'mmmm'.
|
|- [DOCUMENT]                  A directory containing document files
|                              relating to this disk.
|      |
|      |- DOCINFO.TXT          Description of files in the DOCUMENT
|      |                       directory.
|      |
|      |- LOSAPDR.TXT          A machine readable version of the LOSAPDR
|      |                       SIS document describing the format and
|      |                       content of the data files.
|      |
|      |- REPORT.ASC           ASCII text version of REPORT.PS.
|      |
|      |- REPORT.LBL           A PDS detached label describing REPORT.ASC & REPORT.PS.
|      |
|      |- REPORT.PS            A PostScript document that gives specific
|                              information about each LOSAPDR after a
|                              quality check and validation.
|
|- [INDEX]                     A directory containing index files relating
|      |                       to this disk.
|      |
|      |- INDEX.LBL            A PDS detached label describing INDEX.TAB.
|      |
|      |- INDEX.TAB            Tabular summary of data files.
|      |
|      |- INDXINFO.TXT         Description of files in the INDEX directory.
```

5. Recommended CD-ROM Drives and Driver Software

VAX/VMS
Drive: Digital Equipment Corporation (DEC) RRD40 or RRD50. Driver: DEC VFS CD-ROM driver V4.7 or V5.2 and up.

Note: The driver software may be obtained from Jason Hyon at JPL. It is necessary to use this driver to access Extended Attribute Records (XARs) on a CD-ROM.

VAX/Ultrix
Drive: DEC RRD40 or RRD50. Driver: Supplied with Ultrix 3.1.

Note: Internet users can obtain a copy of the "cdio" software package via anonymous ftp from the "space.mit.edu" server in the file named "src/cdio.shar". Contact Dr. Peter Ford at Massachusetts Institute of Technology for details (617-253-6485 or pgf@space.mit.edu).

IBM PC
Drive: Toshiba, Hitachi, Sony, or compatible. Driver: Microsoft MSCDEX version 2.2.

Note:The latest version of MSCDEX (released in February 1990) is generally available. Contact Jason Hyon for assistance in locating a copy.

Apple Macintosh
Drive: Apple CD SC (Sony) or Toshiba. Driver: Apple CD-ROM driver.

Note: The Toshiba drive requires a separate driver, which may be obtained from Toshiba.

Sun Micro (SunOS 4.0.x and earlier)
Drive: Delta Microsystems SS-660 (Sony). Driver: Delta Microsystems driver or SUN sr.o Driver.

Note: For questions concerning this driver, contact Denis Down at Delta Microsystems, 415-449-6881.

Sun Micro (SunOS 4.0.x and later)
Drive: Sun Microsystems. Driver: SunOS sr.o driver.

Note: A patch must be made to SunOS before the Sun driver can access any CD-ROM files containing Extended Attribute Records. A copy of this patch is available to Internet users via anonymous ftp from the "space.mit.edu" server in the file named "src/SunOS.4.x.CD-ROM.patch".

6.Errata and Disclaimer

A cumulative list of anomalies and errors is maintained in the file ERRATA.TXT at the root directory of this volume.

Although considerable care has gone into making this volume, errors are both possible and likely. Users of the data are advised to exercise the same caution as they would when dealing with any other unknown data set.

Reports of errors or difficulties would be appreciated. Please contact one of the persons listed herein.

7. Whom to Contact for Information

For questions concerning this volume set, data products and documentation:

Jim Alexopoulos
Washington University Dept. of Earth and Planetary Sciences
1 Brookings Drive
Campus Box 1169
St. Louis, MO 63130
314-935-5365

Electronic mail address: Internet: jim@wuzzy.wustl.edu

For questions about how to read the CD-ROM:

Jason J. Hyon
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive MS 525-3610
Pasadena, CA 91109
818-306-6054

Electronic mail addresses:
Internet: jhyon@jplpds.jpl.nasa.gov
NASAmail: JHYON
NSI: JPLPDS::JHYON X.400:
(ID:JHYON,PRMD:NASAMAIL,ADMD:TELEMAIL,C:USA)

For questions concerning the generation of LOSAPDR products:

William L. Sjogren
Magellan Gravity Principal Investigator
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive MS 301-150
Pasadena, CA 91109 818-354-4868

```
     Electronic mail address:
            Internet: wls@nomad.jpl.nasa.gov


For questions concerning LOSAPDR data:


     William L. Sjogren
     Jet Propulsion Laboratory Pasadena, CA


     Dr. Roger J. Phillips
     Washington University Dept. of Earth and Planetary Sciences
     1 Brookings Dr. Campus Box 1169
     St. Louis, MO 63130 314-935-6356

     Electronic mail address: Internet: phillips@wustite.wustl.edu

For questions concerning LOSAPDR labels:

     Dr. Richard Simpson
     Stanford University
     Durand Bldg. Room 232
     Stanford, CA 94305-4055
     415-723-3525

     Electronic mail address: Internet: rsimpson@magellan.stanford.edu

This disk was produced by Jim Alexopoulos.
```

D.2 INDXINFO.TXT

Each PDS archive volume shall include an “INDXINFO.TXT” file in the INDEX subdirectory that contains an overview of the contents and structure of the index table or tables on the volume as well as usage notes. An example is provided here as guidance for compiling this file.

D.2.1 Example

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                                = PDS3

RECORD_TYPE                                   = STREAM
OBJECT                                        = TEXT
NOTE                                          = "Notes on using the image index tables."
PUBLICATION_DATE                              = 1990-12-20
END_OBJECT                                    = TEXT
END

NOTES ON USING THE IMAGE INDEX TABLES

These notes describe the contents and format of the two image index tables on this CD-ROM,
INDEX.TAB and CUMINDEX.TAB.

The image index table (INDEX.TAB) contains one record for each image file on this Viking
Orbiter CD-ROM.  The cumulative image index table (CUMINDEX.TAB) contains one record for each
image file on all the Viking Orbiter CD-ROMs published so far.  The following description
applies to both of these tables.

The image index tables are formatted so that they may be read directly into many database
management systems on various computers.

All fields are separated by commas, and character fields are enclosed in double quotation
marks (").  Each record contains 512 bytes of ASCII character data (1 character = 1 byte).
Bytes 511 and 512 contain the ASCII carriage return and line feed characters.  This allows the
table to be treated as a fixed length record file on computers that support this file type and
as a normal text file on other computers. The structure and content of the image index tables
are described in the file VOLINFO.TXT located in the DOCUMENT directory.  The files INDEX.LBL
and CUMINDEX.LBL contain labels for INDEX.TAB and CUMINDEX.TAB coded in the Object Description
Language (ODL), providing a formal description of the index table structure.

Users of most commercial database management systems should be able to use the list below to
define the names and characteristics of each field and then to load the tables into their
systems using a delimited ASCII text input format.  If necessary the specific column start
positions and lengths can be used to load the data.


For personal computer users, DBASE III DBF structures are also provided in the files INDEX.DBF
and CUMINDEX.DBF.  These files can be used to load the INDEX.TAB or CUMINDEX.TAB files into
DBASE III or IV with the following commands:

USE INDEX
APPEND FROM INDEX.TAB DELIMITED

USE CUMINDEX
APPEND FROM CUMINDEX.TAB DELIMITED

Once the table is loaded into DBASE III, it can generally be automatically loaded into other
data managers or spreadsheets that provide search and retrieval capabilities.
```

D.3 SOFTINFO.TXT

Each PDS archive volume that contains software (in the SOFTWARE subdirectory) shall include a "SOFTINFO.TXT" file. This file contains a description of the software and usage information. An outline and example are provided here as guidance for compiling this file.

D.3.1 Outline

I. PDS TEXT Object (must appear in an attached or detached label)

II. Contents

1. Introduction

2. Software Description
   A brief description of software included on the volume. This can be broken down into separate sections for each type of software. This should indicate where the software and its documentation reside in the software hierarchy, as well as describe any known limitations or problems.

3. Software Directory Structure (optional)

4. Software License Information and Disclaimers (if appropriate)

D.3.2 Example

```
PDS_VERSION_ID                          = PDS3

RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 80
OBJECT                                  = TEXT
INTERCHANGE_FORMAT                      = ASCII
PUBLICATION_DATE                        = 1994-10-01
NOTE                                    = "Description of software provided with the
                                           Clementine CD-ROM set"
END_OBJECT                              = TEXT
END

                         Clementine  Software

1. Introduction

This directory contains software that provides display and processing
capabilities for the Clementine data archived on this CD-ROM set.


2. Software Description

2.1. Decompression Software

The PCDOS, MACSYS7 and SUNOS subdirectories all contain software which can be used to
decompress the Clementine raw images.  CLEMDCMP will decompress the raw image and output it
into one of four formats:

    1) decompressed PDS labeled file which contains PDS labels, the histogram object, and an
```

image object, either the browse image or the full image
2) decompressed image file, no labels
3) a decompressed image in the GIF format
4) a decompressed image in the TIFF format

The source code is provided in the SRC subdirectory, of each platform subdirectory. Instructions on how to install and run the software is in the file CLEMDCMP.TXT in the DOC subdirectory, of each platform subdirectory.

Because the image decompression program, CLEMDCMP, requires a Discrete Cosine Transform (DCT) it may take several minutes to decompress an image on hardware platforms with slow processors. For example, in tests on a Macintosh IIci, the decompression takes approximately 4 minutes. CLEMDCMP has been tested on hardware platforms with processors, such as an Intel 486DX2/66-Mhz, and the decompression takes just several seconds.

2.2. Display Software

CLIMDISP in the PCDOS/BIN subdirectory is an image display and processing program. It can be used to display Clementine uncompressed images and histograms. See CLIMDISP.TXT in the PCDOS/DOC subdirectory for instructions on how to install and run the program.

Note: CLIMDISP currently can not create GIF formatted files for the Clementine images. Additionally, it can not read the version of GIF files created by the Clementine Decompression (CLEMDCMP) program which is also included on the Clementine EDR Archive CD-ROMs. If you wish to display Clementine images with CLIMDISP, generate a PDS format image file when decompressing with CLEMDCMP.

A special version of NIH Image, found in the MACSYS7/BIN subdirectory, will display PDS decompressed Clementine images. This program is stored in a Stuffit file which is in BinHex format. See IMAGE.TXT in the DOC subdirectory for instructions on how to install and run the program.

The Clementine EDR image files use the PDS label constructs RECORD_TYPE = "UNK", and ^IMAGE = xxxxx <BYTES> to define the structure of the file. This form of the labels is not supported by the current versions of IMDISP and IMAGE4PDS that are widely distributed by the PDS. To read Clementine decompressed formatted files use the version of IMAGE and CLIMDISP programs that are supplied on this CD-ROM. The Clementine versions CLIMDISP and IMAGE have been tested only on the Clementine data products. No attempt has been made to determine if the Clementine program versions will work on any other PDS data product.

XV is a shareware program for displaying images. XV was written by John Bradley of the University of Pennsylvania. It is in a compressed tar file in the SUNOS/SRC subdirectory. See XV.TXT in the SUNOS/DOC subdirectory for instructions on how to decompress and untar this file. XV will not display PDS labeled files, but will display TIF and GIF formatted files.

The XV software, for image display on a sun/unix environment, is not able to read the Clementine PDS labeled files. If you intend to use XV as the display system for the CLementine data products, output GIF or TIFF images with the CLEMDCMP program.

2.3. SPICE Software

Included on one of the ancillary disks associated with this volume set is the Navigation and Ancillary Information Facility (NAIF) Toolkit and some additional NAIF software. The major component of the NAIF Toolkit is the SPICE Library (SPICELIB), a collection of portable ANSI FORTRAN 77 subroutines. Some of these subroutines are used to read the SPICE kernel files containing Clementine ancillary data, such as spacecraft position, spacecraft attitude, instrument orientation and target body size, shape and orientation. Other SPICELIB subroutines may be used to compute typical observation geometry parameters--such as range, lighting angles, and LAT/LON of camera optic axis intercept on the target body. Several utility programs and SPICELIB demonstration programs are also included in the Toolkit. Versions of this software tested on many popular platforms are provided, as are instructions for porting the code to additional platforms. The FORTRAN subroutines can be called from a user's own application program, whether written in FORTRAN or C, or possibly yet another language. Consult your compiler's Reference Manual for instructions. One of the NAIF programs included in this software collection is PICGEO (for Picture Geometry). It was used to compute all of the geometric parameters appearing in the image labels and index tables. It is included so that users may clearly see the algorithms used in computing these quantities, and so that recalculation of image label geometry parameters using revised algorithms, or adding additional parameters, can be easily achieved.

2.4. Miscellaneous Image Processing Software

MSHELL is an interactive command line and menu driven Image and Signal processing language, developed by ACT Corp., which runs under the Microsoft Windows 3.x or Microsoft NT. MSHELL provides powerful scientific image and signal visualization and processing. A number of custom features were added to the MSHELL Image/Signal Processing Environment to support the Clementine Program. This software is included on one of the ancillary disks associated with this volume set, and will be under a subdirectory of the PCDOS directory.

3. Software Directory Hierarchy

The SOFTWARE subdirectories are based on hardware platforms. Under each platform subdirectory, the executables are in the BIN subdirectory, the source is in the SRC subdirectory and documentation on each program is in the DOC subdirectory. Each DOC subdirectory contains a file, SWINV.CAT which is part of the PDS Software Inventory describing software available within the Planetary Science Community. The contents of the SOFTWARE directory are shown below.

```
[SOFTWARE]
|
|-SOFTINFO.TXT
|
|-[PCDOS]
|   |
|   |-[BIN]
|   |    |
|   |    |-CLEMDCMP.EXE
|   |    |-CLIMDISP.EXE
|   |    |-CLIMDISP.HLP
|   |
|   |-[SRC]
|   |    |
|   |    |-CLEMDCMP.C
|   |    |-PDS.C
|   |    |-BITSTRM.C
|   |    |-DECOMP.C
|   |    |-HUFFMAN.C
|   |    |-WRITEGIF.C
|   |    |-PDS.H
|   |    |-JPEG_C.H
|   |    |-CLEMDCMP.MAK
|   |
|   |-[DOC]
|        |
|        |-CLEMDCMP.TXT
|        |-CLIMDISP.TXT
|        |-SWINV.CAT
|
|-[MACSYS7]
|   |
|   |-[BIN]
|   |   |
|   |   |-CLEMDEXE.HQX
|   |   |-IMAGE.HQX
|   |
|   |-[SRC]
|   |   |
|   |   |-CLEMDSRC.HQX
|   |
|   |-[DOC]
|       |
|       |-CLEMDCMP.TXT
|       |-IMAGE.TXT
|       |-SWINV.CAT
|
|-[SUNOS]
    |
    |-[BIN]
    |   |
    |   |-CLEMDEXE.TZU
    |
    |-[SRC]
    |   |
    |   |-CLEMDSRC.TZU
    |   |-XV3A.TZ
```

```
|
|-[DOC]
   |
   |-CLEMDCMP.TXT
   |-XV.TXT
   |-SWINV.CAT
```

D.4 VOLDESC.CAT

This file contains the VOLUME object, which provides a high-level description of the contents of an archive volume.

The VOLDESC.CAT file must be present at the root level of the archive volume. If the archive volume contains logical volumes, then a VOLDESC.CAT file must be present at the root level of each logical volume. See Section A.28 for more information on using the VOLUME object and for examples of the VOLDESC.CAT file.

(This page intentionally left blank.)

Appendix E. NAIF Toolkit Directory Structure

This appendix contains the software directory structure of the NAIF Toolkit for a SUN. It is an example of a platform-based model for a single platform. Note that the directory organization shown here does not strictly conform to the recommendations discussed in the *Volume Organization and Naming* chapter of this document.

Chapter Contents

Appendix E. NAIF Toolkit Directory Structure E-1

E.1 NAIF Directory E-2
E.2 TOOLKIT Directory E-3
E.3 Using the NAIF Toolkit E-12
E.4 NAIF's File Naming Conventions E-13

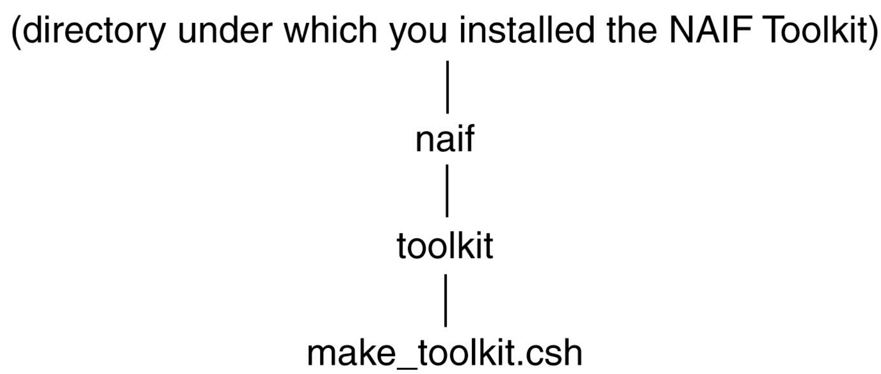
E.1 NAIF Directory

The NAIF directory contains one subdirectory, TOOLKIT. The TOOLKIT tree contains all of the files that make up the NAIF Toolkit.

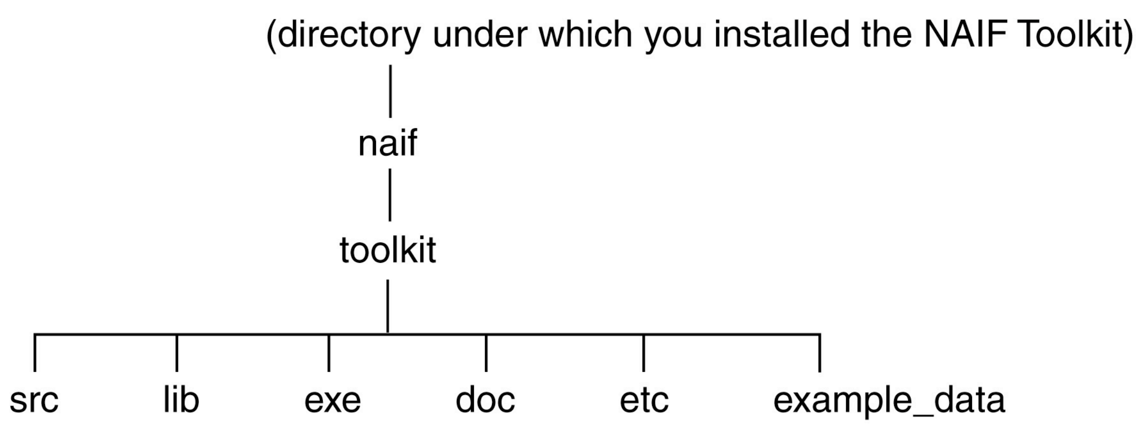
```
(directory under which you installed the NAIF Toolkit)
                          |
                         naif
                          |
                        toolkit
```

E.2 TOOLKIT Directory

The TOOLKIT directory contains the file *make_toolkit.csh*. This is a C shell script that builds all of the object libraries and executables in the TOOLKIT.



TOOLKIT also contains several subdirectories that will be described in more detail in the following sections.



1. SRC
The subdirectories of this directory contain all of the source code for the products in the TOOLKIT.

2. LIB
This directory contains all of the TOOLKIT object libraries.

3. EXE
This directory contains all of the TOOLKIT executables, and where applicable, scripts to run the executables.

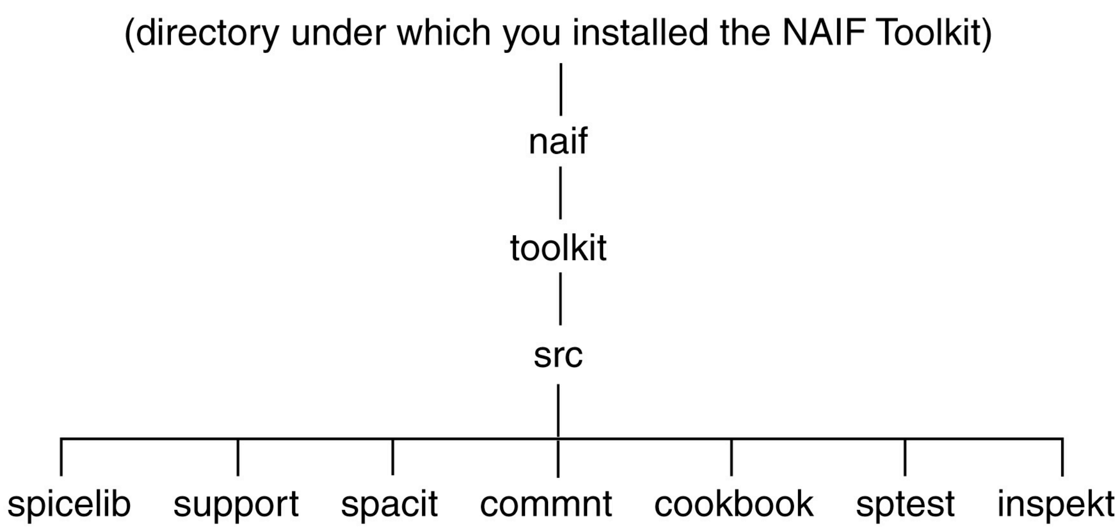
4. DOC
This directory contains all of the TOOLKIT documentation. This includes user's guides for the programs, "Required Reading" files for SPICELIB, documents describing the contents of SPICELIB such as the "Permuted Index and Module Summary," and documents describing the contents and installation of the Toolkit.

5. ETC
The subdirectories of this directory contain product-specific files that are neither source, documentation, nor data. These include configuration files, set up files, and help files. The subdirectory build contains the C shell script that creates the toolkit object libraries and executables.

6. EXAMPLE_DATA
This directory contains example data for use with the cookbook and *sptest* programs. These files are to be used only with these programs.

E.2.1 SRC

The SRC directory contains one subdirectory for each product in the NAIF Toolkit. Each of these product directories contains the source code files and procedures to create the executable or object library.



E.2.1.1 SPICELIB

SPICELIB is a Fortran source code library that contains approximately 650 functions, subroutines, and entry points.

This directory contains the SPICELIB source files.

```
(directory under which you installed the NAIF Toolkit)
                    |
                   naif
                    |
                  toolkit
                    |
                   src
                    |
                 spicelib

                   *.f
```

E.2.1.2 SUPPORT

SUPPORT is a Fortran source code library that contains routines that support the Toolkit programs. These routines are not intended to be used by anyone except NAIF. These routines are not officially supported and may undergo radical changes such as calling sequence changes. They may even be deleted. Do not use them!

This directory contains the SUPPORT library source files.

```
(directory under which you installed the NAIF Toolkit)
                    |
                   naif
                    |
                  toolkit
                    |
                   src
                    |
                 support

                   *.f
```

E.2.1.3 SPACIT

Spacit is a utility program that performs three functions: it converts transfer format SPK, CK and EK files to binary format, it converts binary SPK, CK and EK files to transfer format, and it summarizes the contents of binary SPK, CK and EK files.

This directory contains the source code for the *spacit* main program and supporting routines.

```
(directory under which you installed the NAIF Toolkit)
                          |
                        naif
                          |
                       toolkit
                          |
                         src
                          |
                       spaclit

                     spaclit.main
                         *.f
```

E.2.1.4 COMMNT

Commnt is a utility program that is used to add comments, extract comments, read comments, or delete comments in SPICE SPK, CK and EK files.

This directory contains the *commnt* main program source file.

```
(directory under which you installed the NAIF Toolkit)
                          |
                        naif
                          |
                       toolkit
                          |
                         src
                          |
                        commnt

                     commnt.main
```

E.2.1.5 COOKBOOK

The COOKBOOK programs are sample programs that demonstrate how to use SPICELIB routines to obtain state vectors, convert between different time representations, manipulate the comments in binary SPK and CK files, and solve simple geometry problems.

This directory contains the COOKBOOK program source files.

```
(directory under which you installed the NAIF Toolkit)
                          |
                        naif
                          |
                       toolkit
                          |
                         src
                          |
                      cookbook

                     fstspk.main
                     simple.main
                     states.main
                     subpt.main
                     tictoc.main
```

E.2.1.6 INSPEKT

Inspekt is a program that allows you to examine the contents of an events component of an E-kernel.

This directory contains the source code for the *inspekt* main program and supporting routines.

```
(directory under which you installed the NAIF Toolkit)
                         |
                        naif
                         |
                       toolkit
                         |
                        src
                         |
                       inspekt

                    inspekt.main
                    *.f
                    *.inc
```

E.2.1.7 SPTEST

Sptest is a utility program that tests the SPK file readers by comparing states read on the NAIF VAX with states read on the target machine.

This directory contains the *sptest* program source file.

```
(directory under which you installed the NAIF Toolkit)
                         |
                        naif
                         |
                       toolkit
                         |
                        src
                         |
                       sptest

                    sptest.main
```

E.2.2 LIB

The LIB directory contains *spicelib.a*, the object library for SPICELIB. It also contains the object library *support.a*, but this library is for use by the Toolkit programs only. Do not link your applications with it!

```
(directory under which you installed the NAIF Toolkit)
                    |
                   naif
                    |
                  toolkit
                    |
                   lib

                spicelib.a
                support.a
```

E.2.3 EXE

The EXE directory contains the NAIF Toolkit executables and, where applicable, scripts to run executables.

```
(directory under which you installed the NAIF Toolkit)
                    |
                   naif
                    |
                  toolkit
                    |
                   exe

                 commnt
                 fstspk
                 inspekt
                 simple
                 spacit
                 sptest
                 states
                 subpt
                 tictoc
```

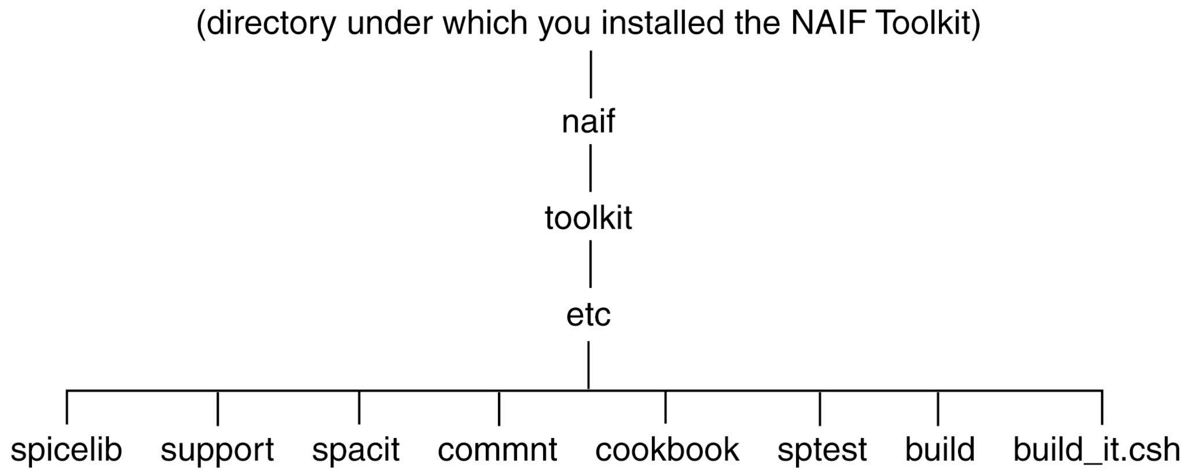
E.2.4 DOC

The DOC directory contains all of the TOOLKIT documentation that is available on-line. This includes the user's guides for the programs, all "Required Reading" files for SPICELIB, all documents describing the contents and porting of SPICELIB, and documents describing the installation and contents of the Toolkit. Please note that the INSPEKT user's guide is not available on-line.

```
(directory under which you installed the NAIF Toolkit)
                          |
                        naif
                          |
                       toolkit
                          |
                         doc

                     commnt.ug
                     fstspk.ug
                     simple.ug
                     spacit.ug
                     sptest.ug
                     states.ug
                     subpt.ug
                     tictoc.ug
                     *.req
                     category.txt
                     libsum.txt
                     permuted_index.txt
                     porting.txt
                     toolkit_install.txt
                     toolkit_description.txt
```

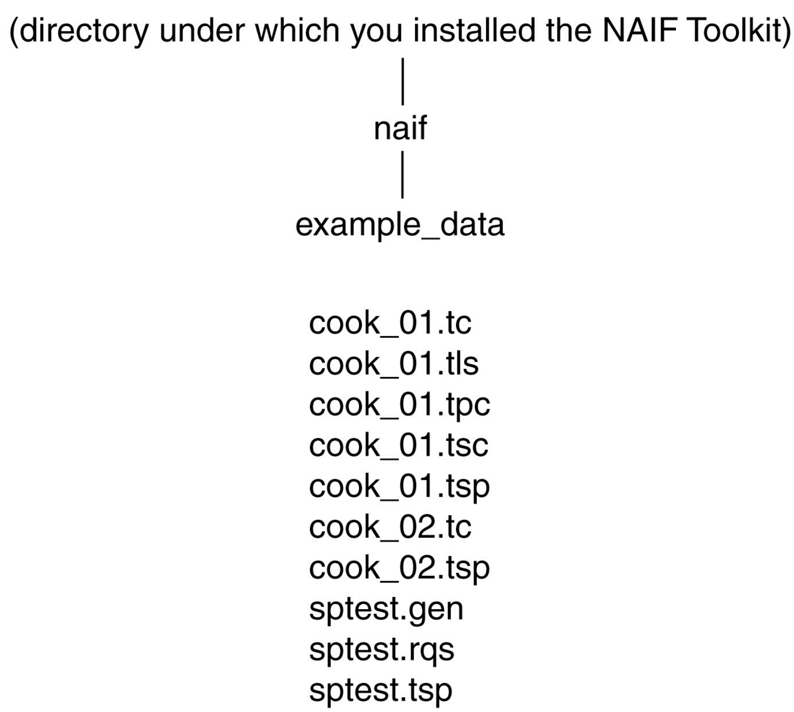
E.2.5 ETC

The ETC directory contains all files for the Toolkit products that are not source, documentation, or data such as set up files, configuration files or help files. It also contains the C shell script used to build the toolkit object libraries and executables.



E.2.6 EXAMPLE_DATA

The EXAMPLE_DATA directory contains all of the NAIF Toolkit data. This data are intended only to be used with the TOOLKIT programs, and are included only to help you get started using the Toolkit.



E.3 Using the NAIF Toolkit

After the installation has been completed successfully, there are a few things that you need to do to get started using SPICELIB. We recommend that you print out the source code for the cookbook programs (*./naif/toolkit/src/cookbook/*.main*) and examine it. Try running some of the cookbook programs yourself. The cookbook programs demonstrate how to use SPICELIB routines to obtain state vectors, convert between different time representations, manipulate the comments in binary SPK and CK files, and solve simple geometry problems.

Once you're ready to get your hands dirty, you should read the required reading files for SPICELIB. The required reading files are located in the directory *./naif/toolkit/doc* and have the extension ".req". They are text files that describe families of subroutines and how they interact with the rest of SPICELIB.

The most important required reading files are: TIME, KERNEL, SPK, CK, SCLK, SPC, and NAIF_IDS. You should read at least these.

After you've done these things, you're ready to start programming with SPICELIB!

E.4 NAIF's File Naming Conventions

NAIF follows a set of conventions for naming files based on the contents of the files. This allows you to find certain types of files in a directory tree quickly. The following table lists the current naming conventions.

*.for, *.f	Fortran-77 source code files
*.main	Source code files for program modules
*.inc	Fortran-77 include files
*.c	C source code files
*.o	Unix object files
*.obj	VAX/VMS object files
*.a	Unix object library files
*.olb	VAX/VMS object library files
*.tsp	Transfer format SPK (ephemeris) files
*.bsp	Binary format SPK (ephemeris) files
*.tc	Transfer format CK (pointing) files
*.bc	Binary format CK (pointing) files
*.ti	Text IK (instrument parameters) files
*.tls	Leapseconds kernel files
*.tpc	Physical and cartographic constants kernel files
*.tsc	Spacecraft clock coefficients kernel files
*.txt	Text format documentation files
*.ug	Text format User's Guides
*.req	Text format SPICELIB Required Reading files
make_toolkit.csh, build_it.csh	Unix C shell script files for creating the toolkit object libraries and executables
make_toolkit.sh, build_it.sh	Unix Bourne shell script files for creating the toolkit object libraries and executables
(product name)	Unix executable files. For example, spacit is the executable file for the product spacit
make_(product name).com	VAX/VMS command procedures for creating products. For example, make_spicelib.com creates the object library spicelib.olb, while make_spacit.com creates the executable spacit.exe.
(product name).exe	VAX/VMS executable files. For example, spacit.exe is the executable file for the product spacit.

These conventions are preliminary. As coordination with AMMOS and the Planetary Data System (PDS) occurs, these conventions may be revised.

(This page intentionally left blank.)

Appendix F. Acronyms

AGU	American Geophysical Union
AMMOS	Advanced Multi-Mission Operations System
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
BNF	Backus-Naur Form
CCSDS	Consultative Committee on Space Data Systems
CDROM	Compact Disk Read-Only Medium
CD-WO	Compact Disk –Write Once
CODMAC	Committee on Data Management and Computation
CR/LF	Carriage Return/Line Feed (ASCII record delimiters)
DA	Data Administrator
DAT	Digital Audio Tape
DBA	Database Administrator
DE	Data Engineer
DIM	Digital Image Model
DN	Discipline Node
DTM	Digital Terrain Model
DVD(ROM)	Digital Video Disk (Read-Only Medium)
EBCDIC	Extended Binary Coded Decimal Interchange Code
ECR	Engineering Change Request
EDR	Experiment Data Records
ET	Ephemeris Time
GIF	Graphics Interface Format
GSFC	Goddard Space Flight Center (Greenbelt, Maryland)
HTML	Hyper-Text Markup Language
IAU	International Astronomical Union
IDS	Inter-Disciplinary Scientist
IRTM	Infrared Thermal Mapper
ISO	International Standards Organization
JD	Julian Date
JPL	Jet Propulsion Laboratory (Pasadena, California)
LSB	Least Significant Byte (or Bit) first
MSB	Most Significant Byte (or Bit) first
NAIF	Navigation and Ancillary Information Facility
NASA	National Aeronautics and Space Administration
NBS	National Bureau of Standards (now NIST)
NIST	National Bureau of Standards and Technology (formerly NBS)
NSI/DECNET	NASA Science Internet/DEC Network
NSSDC	National Space Science Data Center
ODL	Object Description Language

PCWG	Planetary Cartography Working Group
PDS	Planetary Data System
PI	Principal Investigator
PNG	Portable Network Graphics
PSDD	Planetary Science Data Dictionary
PVL	Parameter Value Language
RPIF	Regional Planetary Image Facility
SEDR	Supplementary Experimental Data Record
SFDU	Standard Formatted Data Unit
SGML	Standard Generalized Markup Language
SI	Systeme Internationale d'Unite– (International Standard for Units)
SIS	System Interface Specification
SPICE	Spacecraft, Planet, Instrument C-Matrix Events
SQL	Structured Query Language
UDF	Universal Disc Format
URL	Uniform Resource Locator
UTC	Universal Time, Coordinated
WORM	Write Once, Read Many (optical disk)
XAR	Extended Attribute Record

Appendix G. SAVED Data

In rare cases data will be encountered in the PDS archive which are classified as having ARCHIVE_STATUS = SAVED. These data are being preserved in a primitive form either pending the production of an archive-quality product, or as part of a save-the-bits campaign for a defunct mission or project. In these cases the available information has been preserved in as close an approximation of PDS archive format as possible. Following is a description of the criteria applied to datasets considered for safing, and the PDS procedures applied during the process. It is provided here for information only – saved datasets are not considered acceptable for the purposes of meeting PDS archiving requirements.

G.1 Safekeeping Process and Procedures

The decision to save a dataset will normally be made by the discipline node after discussion with the data provider. The decision may also be made by a data engineer at the Central Node after discussion with both the data provider and the most relevant discipline node(s). Preservation should take place according to the following procedures.

1. The details of every dataset to be saved will be discussed in a conference, such as a telecon or iteration of e-mail messages, among the data provider, the representatives of the relevant discipline node(s), and a data engineer from Central Node. This discussion will address:

    a. The characteristics of the data to be preserved
    b. The reasons for preserving rather than archiving the data
    c. The timetable for producing an archival product from the preserved data
    d. The proposed unique VOLUME_ID for the product
    e. The extent of the additional information to be included

2. The conclusions of the decision-making conference will be summarized by the data engineer and distributed to all participants.

3. The dataset will normally be prepared and delivered by the data provider according to the agreed content and format.

4. The data engineer at the Central Node will ensure that the product is incorporated into the Distributed Inventory System (DIS).

G.2 Safekeeping Standards

The following items are desirable for any preserved dataset. Some are required, as noted.

1. VOLUME_ID – This ID is required for every preserved product, must be unique within PDS, and must conform to the volume naming standards of PDS.

2. DIS.LBL – This label file is required for every preserved product and must conform to the PDS labeling standards.

3. AAREADME.TXT – This file describes the directory structure and the content of the volume. It also includes contact information for the original source of the data.

4. INDEX.TXT – This file is used if the individual data files do not have PDS labels. It consists of free format text and is a less rigorous version of INDEX.TAB.

5. Minimal labels – Individual files should be labeled with "minimal labels" as described in Section 5.2.3.

6. Document directory – This directory is optional but all files must have minimal labels.

7. Software directory – This directory is optional but all files must have minimal labels

Items 1 and 2 are absolutely required for all preserved datasets. Items 3 through 5, though not required, are strongly recommended for every preserved dataset. Items 6 and 7 are strongly recommended where appropriate.

Appendix H. PDS Data Group Definitions

This section provides an alphabetical reference of approved PDS data group definitions used in labeling data objects. The definitions include descriptions, lists of required and optional keywords, and one or more examples of specific groups. For a more detailed discussion on data groups, see the *Data Objects / Groups* chapter in this document.

Data group definitions are refined and augmented from time to time, as user community needs arise, so object definitions for products designed under older versions of the Standards may differ significantly. To check the current state of any object or group definition, consult a PDS data engineer or either of these URLs:

PDS Catalog Search: **http://pdsproto.jpl.nasa.gov/onlinecatalog/top.cfm**

Data Dictionary Search: **http://pdsproto.jpl.nasa.gov/ddcolstdval/newdd/top.cfm**

The examples provided in this Appendix are based on both existing and planned PDS archive products, modified to reflect the current version of the PDS Standards. Additional examples may be obtained by contacting a PDS Data Engineer.

NOTE: Any keywords in the *Planetary Science Data Dictionary* may also be included in a specific data group definition.

Chapter Contents

Appendix H. PDS Data Group Definitions H-1

H.1 BAND_BIN H-3
H.2 BAND_SUFFIX H-4
H.3 LINE_SUFFIX H-5
H.4 PARAMETERS H-6
H.5 SAMPLE_SUFFIX H-7

H.1 BAND_BIN

The BAND_BIN group provides a mechanism for grouping keywords that describe the properties of each "bin" along a spectral axis. It is primarily designed for use within the SPECTRAL_QUBE object.

See Appendix A.25 for a detailed description of the SPECTRAL_QUBE.

H.1.1 Required Keywords

1. BANDS
2. BAND_BIN_CENTER
3. BAND_BIN_UNIT
4. BAND_BIN_WIDTH

H.1.2 Optional Keywords

1. BAND_BIN_STANDARD_DEVIATION
2. BAND_BIN_DETECTOR
3. BAND_BIN_GRATING_POSITION
4. BAND_BIN_ORIGINAL_BAND
5. BAND_BIN_BAND_NUMBER
6. BAND_BIN_FILTER_NUMBER
7. BAND_BIN_BASE
8. BAND_BIN_MULTIPLIER

H.1.3 Example

The following label fragment shows the BAND_BIN group:

```
GROUP                        = BAND_BIN
  BANDS                      = 3
  BAND_BIN_UNIT              = MICROMETER
  BAND_BIN_FILTER_NUMBER     = (1, 2, 3)
  BAND_BIN_BAND_NUMBER       = (2, 3, 4)
  BAND_BIN_CENTER            = (6.78, 9.35, 14.88)
  BAND_BIN_WIDTH             = (1.01, 1.20, 0.87)
  BAND_BIN_BASE              = (0.0, 0.0, 0.0)
  BAND_BIN_MULTIPLIER        = (1.0, 1.0, 1.0)
END_OBJECT                   = BAND_BIN
```

H.2 BAND_SUFFIX

The BAND_SUFFIX group provides a mechanism for grouping keywords that describe the properties of each BAND Suffix plane, or BACKPLANE, of a SPECTRAL_QUBE.

See Appendix A.25 for a detailed description of the SPECTRAL_QUBE.

H.2.1 Required Keywords

1. SUFFIX_NAME
2. SUFFIX_ITEM_BYTES
3. SUFFIX_ITEM_TYPE

H.2.2 Optional Keywords

1. SUFFIX_BASE
2. SUFFIX_MULTIPLIER
3. SUFFIX_VALID_MINIMUM
4. SUFFIX_NULL
5. SUFFIX_LOW_REPR_SAT
6. SUFFIX_LOW_INSTR_SAT
7. SUFFIX_HIGH_REPR_SAT
8. SUFFIX_HIGH_INSTR_SAT
9. SUFFIX_UNIT
10. BIT_MASK

H.2.3 Example

The following label fragment shows the BAND_SUFFIX group:

```
GROUP                       = BAND_SUFFIX
  SUFFIX_NAME               = (LATITUDE, LONGITUDE)
  SUFFIX_UNIT               = (DEGREE, DEGREE)
  SUFFIX_ITEM_BYTES         = (4, 4)
  SUFFIX_ITEM_TYPE          = (IEEE_REAL, IEEE_REAL)
  SUFFIX_BASE               = (0.000000, 0.000000)
  SUFFIX_MULTIPLIER         = (1.000000, 1.000000)
END_OBJECT                  = BAND_SUFFIX
```

H.3 LINE_SUFFIX

The LINE_SUFFIX group provides a mechanism for grouping keywords that describe the properties of each LINE Suffix plane, or BOTTOMPLANE, of a SPECTRAL_QUBE.

See Appendix A.25 for a detailed description of the SPECTRAL_QUBE.

H.3.1 Required Keywords

1. SUFFIX_NAME
2. SUFFIX_ITEM_BYTES
3. SUFFIX_ITEM_TYPE

H.3.2 Optional Keywords

1. SUFFIX_BASE
2. SUFFIX_MULTIPLIER
3. SUFFIX_VALID_MINIMUM
4. SUFFIX_NULL
5. SUFFIX_LOW_REPR_SAT
6. SUFFIX_LOW_INSTR_SAT
7. SUFFIX_HIGH_REPR_SAT
8. SUFFIX_HIGH_INSTR_SAT
9. SUFFIX_UNIT
10. BIT_MASK

H.3.3 Example

The following label fragment shows the LINE_SUFFIX group:

```
GROUP                           = LINE_SUFFIX
  SUFFIX_NAME                   = VERTICAL_DESTRIPE
  SUFFIX_ITEM_BYTES             = 4
  SUFFIX_ITEM_TYPE              = IEEE_REAL
  SUFFIX_BASE                   = 0.000000
  SUFFIX_MULTIPLIER             = 1.000000
  SUFFIX_VALID_MINIMUM          = 16#FFEFFFFF#
  SUFFIX_LOW_REPR_SAT           = 16#FFFEFFFF#
  SUFFIX_LOW_INSTR_SAT          = 16#FFFDFFFF#
  SUFFIX_HIGH_REPR_SAT          = 16#FFFBFFFF#
  SUFFIX_HIGH_INSTR_SAT         = 16#FFFCFFFF#
END_OBJECT                      = LINE_SUFFIX
```

H.4 PARAMETERS

The PARAMETERS group provides a mechanism for grouping multiple sets of related parameters within a data product label. An alias, PARMS, exists for the PARAMETERS group.

See Chapter 13 of the Standards Reference for a complete description of GROUPs.

H.4.1 Required Keywords

None

H.4.2 Optional Keywords

1. psdd

H.4.3 Example

The following label fragment shows the PARAMETERS group:

```
GROUP                       = COMMANDED_INST_PARAMETERS
 SHUTTER_MODE               = "BOTSIM"
 FILTER_NUMBER              = 5
 FILTER_NAME                = "L570-R570"
 EXPOSURE_DURATION          = 1.05
END_OBJECT                  = COMMANDED_INST_PARAMETERS

GROUP                       = TELEMETRY_INST_PARAMETERS
 SHUTTER_MODE               = "AUTO"
 FILTER_NUMBER              = 0
 FILTER_NAME                = "CLEAR"
 EXPOSURE_DURATION          = 0.773
END_OBJECT                  = TELEMETRY_INST_PARAMETERS

GROUP                       = TELEMETRY_PARMS
 SHUTTER_MODE               = "AUTO"
 FILTER_NUMBER              = 0
 FILTER_NAME                = "CLEAR"
 EXPOSURE_DURATION          = 0.773
END_OBJECT                  = TELEMETRY_PARMS
```

H.5 SAMPLE_SUFFIX

The SAMPLE_SUFFIX group provides a mechanism for grouping keywords that describe the properties of each SAMPLE Suffix plane, or SIDEPLANE, of a SPECTRAL_QUBE.

See Appendix A.25 for a detailed description of the SPECTRAL_QUBE.

H.5.1 Required Keywords

1. SUFFIX_NAME
2. SUFFIX_ITEM_BYTES
3. SUFFIX_ITEM_TYPE

H.5.2 Optional Keywords

1. SUFFIX_BASE
2. SUFFIX_MULTIPLIER
3. SUFFIX_VALID_MINIMUM
4. SUFFIX_NULL
5. SUFFIX_LOW_REPR_SAT
6. SUFFIX_LOW_INSTR_SAT
7. SUFFIX_HIGH_REPR_SAT
8. SUFFIX_HIGH_INSTR_SAT
9. SUFFIX_UNIT
10. BIT_MASK

H.5.3 Example

The following label fragment shows the SAMPLE_SUFFIX group:

```
GROUP                          = SAMPLE_SUFFIX
  SUFFIX_NAME                  = HORIZONTAL_DESTRIPE
  SUFFIX_ITEM_BYTES            = 4
  SUFFIX_ITEM_TYPE             = IEEE_REAL
  SUFFIX_BASE                  = 0.000000
  SUFFIX_MULTIPLIER            = 1.000000
  SUFFIX_VALID_MINIMUM         = 16#FFEFFFFF#
  SUFFIX_NULL                  = 16#FFFFFFFF#
  SUFFIX_LOW_REPR_SAT          = 16#FFFEFFFF#
  SUFFIX_LOW_INSTR_SAT         = 16#FFFDFFFF#
  SUFFIX_HIGH_REPR_SAT         = 16#FFFBFFFF#
  SUFFIX_HIGH_INSTR_SAT        = 16#FFFCFFFF#
END_OBJECT                     = SAMPLE_SUFFIX
```

(This page intentionally left blank.)

Appendix I. Data Compression Formats

This appendix provides a brief description of each of the compression formats that has been approved by the PDS for archive data.

Each section in this appendix includes a high level description of the compression format, PDS-specific implementation rules, and information about how to properly label files implementing the compression algorithm. Each section should also include a sample label.

Chapter Contents

Appendix I. Data Compression Formats I-1

I.1 CLEM-JPEG I-3
I.2 HUFFMAN FIRST DIFFERENCE I-4
I.3 JPEG 2000 .... I-5
I.4 PREVIOUS PIXEL I-10
I.5 RUN LENGTH I-11
I.6 ZIP I-12

I.1 CLEM-JPEG

TBD

I.1.1 PDS Implementation Rules

TBD

I.1.2 Labeling

TBD

I.1.3 Label Example

TBD

I.2 HUFFMAN FIRST DIFFERENCE

TBD

I.2.1 PDS Implementation Rules

TBD

I.2.2 Labeling

TBD

I.2.3 Label Example

TBD

I.3 JPEG 2000

JPEG 2000 is defined as an "image coding system". The ISO/IEC specification describing it includes not only the syntax for a compressed image codestream (mime type "J2C"), but also a description of the binary "JP2" file format that may be used to enhance the utility of the codestream.

Unlike many older compression algorithms, JPEG 2000 provides a great deal of flexibility in the way in which data may be stored in the codestream and retrieved from it. This flexibility allows for the progressive decompression of "layers" of the image with increasing resolution or precision. It also permits the extraction and decompression of only a portion or "tile" of the image. Specific portions of the image of particular interest to the intended audience may also be stored at the beginning of the codestream so that they may be accessed and decompressed first. (This would be of potential interest for approach images where the target of the observation fills only a small portion of the field of view.)

All of the information necessary to successfully decompress a JPEG 2000 image is contained in the J2C codestream. However, the information necessary to take advantage of the additional capabilities is only available in the JP2 format.

A JP2 file essentially consists of a set of "boxes" that encapsulate both the J2C codestream and the meta data that describe it (Figure I.1). The first two of these boxes provide information that identifies the file as a JP2 formatted file. The following "superbox" is the JP2 header box which contains information about the image size, resolution, colorspace, etc. Following this, in no particular order, are contiguous codestream boxes containing the compressed image data and (optionally) intellectual property rights boxes, XML boxes containing vendor-specific meta data, and UUID boxes containing reference URLs. In this document, all of these non-image boxes will be collectively referred to as the "JP2 binary wrapper."

PDS requires the presence of the JP2 binary wrapper so that external software may take full advantage of the JPEG 2000 capabilities. PDS software will have the capability to fully decompress the entire data file, but will not necessarily have the capability to decompress subsets of the codestream such as individual resolution layers or tiles.

The ISO/IEC specification defining JPEG 2000 is entitled "Information technology – JPEG 2000 image coding system" and may be ordered from the ISO by going to their web site (http://www.iso.ch/) and searching on "JPEG 2000".

I.3.1 Table of Compression Ratios

The JPEG 2000 compression algorithm was tested on two Mars Express HRSC images. In both cases, the binary headers and line prefix information on these images were retained in order to provide some additional stress testing of the compression algorithm. With the inclusion of this artificially included binary noise, both of the following images cover the full 16-bit data range.

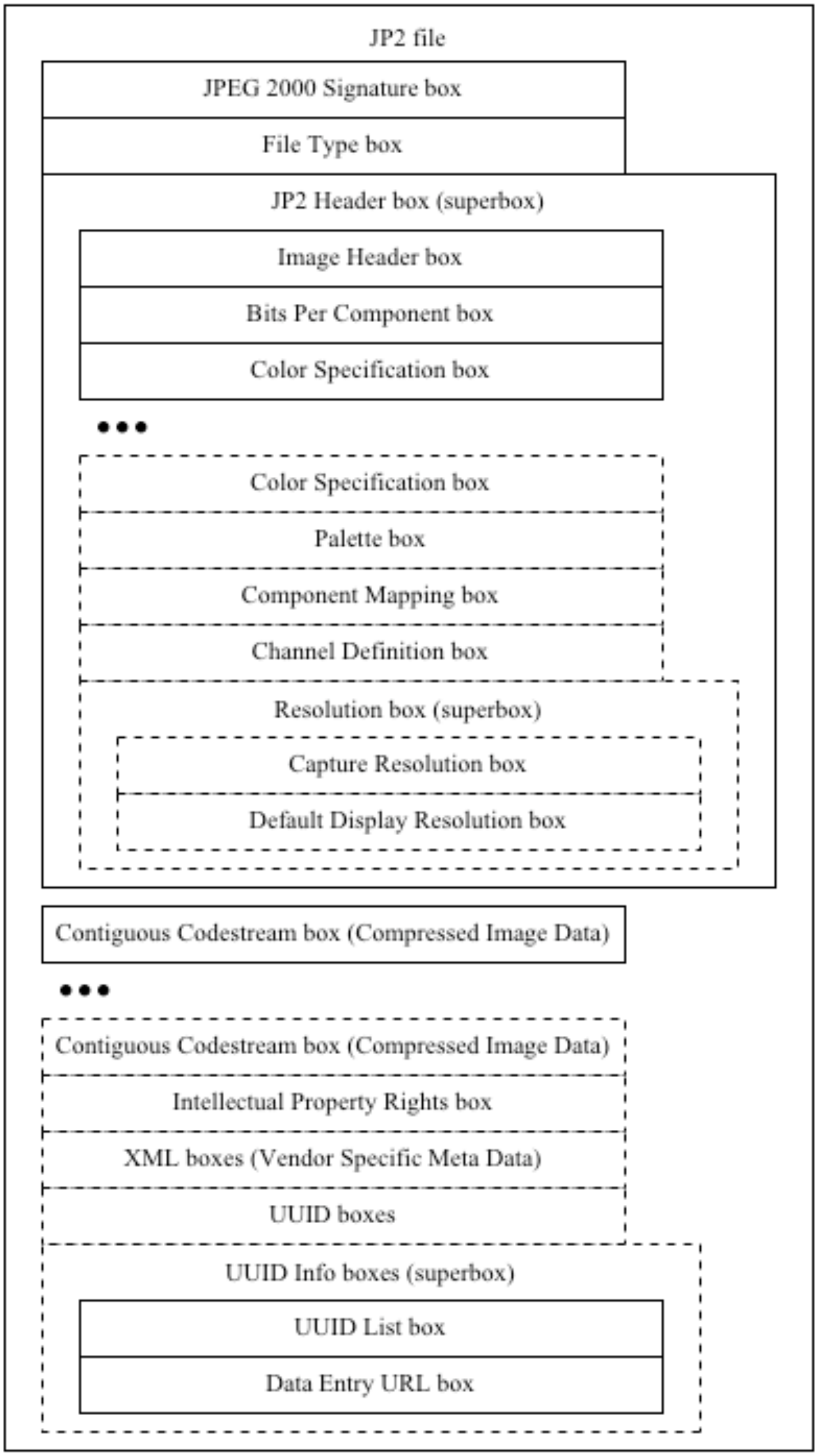


Figure I.1 – Graphical representation of a JP2 file. Dashed lines indicate optional boxes. (Modified from ISO/IEC 15444-1:2004, "Information technology – JPEG 2000 image coding system: Core coding system", figure T.800_FI.1)

The first image, "h0068_0000_s22.img", has 2,618 samples and 119,757 lines, with 16 bit pixels. The image data alone have a dynamic range of 67 to 308 DN, with a standard deviation of 52.5.

The second image, "h0068_0009_s22.img", has 2,618 samples and 11,013 lines, with 16 bit pixels. The image data alone have a dynamic range of 62 to 188 DN, with a standard deviation of 9.2

Both images were also converted to 8 bit data for comparison purposes. Selecting for various tile sizes in the JPEG 2000 compression software, the following lossless compression ratios were obtained:

	h0068_0000_s22		**h0068_0009_s22**	
tile size	16-bit	8-bit	16-bit	8-bit
1024	5.83	2.89	6.00	2.25
512	5.82	2.88	5.99	2.25
256	5.78	2.87	5.94	2.24
128	5.64	2.81	5.80	2.20

These results can be compared with a compression ratio of 3.8 for both of the 16-bit versions of the h0068_0000_s22 and h0068_0009_s22 images, when using Zip compression.

Re-projected images containing large areas of no data and half word image data which do not utilize a full 16-bit data range should result in even higher compression ratios.

I.3.2 PDS Implementation Rules

The JPEG 2000 compression algorithm must be implemented on PDS archive volumes as a codestream encapsulated by the JP2 binary wrapper (ie., not as a bare codestream).

Only lossless compression may be used.

Furthermore, the syntax and features of the compressed codestream must conform to part 1, the "Core coding system", of the ISO/IEC specification defining JPEG 2000, namely 15444-1.

Use of the JPEG 2000 compression algorithm and format is restricted to derived image data where the source, or EDR, products have been archived in an uncompressed format.

I.3.3 Labeling

For each image archived in JPEG 2000 format, two files need to be considered: (1) the compressed file physically included in the archive, and (2) the dynamically generated data file produced by decompressing the JP2 file. These two files have the same name but different extensions: ".JP2" for a JP2 formatted file and "IMG" for the decompressed file. (The "JP2" file extension is reserved exclusively, and must be used, for JPEG 2000 compressed files within the PDS).

Like all PDS data files, both the compressed and the decompressed data files require labels. Both files must be described by a single, detached PDS label file using the combined-detached label approach (see Section 5.2.2). Attached labels are not permitted for JPEG 2000 compressed data, because an attached PDS header would violate the JP2 format. In a combined-detached label, each individual file is described using an explicit FILE object. The general framework is:

```
PDS_VERSION_ID                = PDS3
DATA_SET_ID                   = ...
PRODUCT_ID                    = ...
  (other parameters relevant to both compressed and decompressed files)

OBJECT                        = COMPRESSED_FILE
  (parameters describing the compressed file)
END_OBJECT                    = COMPRESSED_FILE

OBJECT                        = UNCOMPRESSED_FILE
  (parameters describing the uncompressed file)
END_OBJECT                    = UNCOMPRESSED_FILE
END
```

The compressed file is described by a “minimal label” (see Section 5.2.3), and the following keywords are required:

```
FILE_NAME                     = name of the compressed file
RECORD_TYPE                   = UNDEFINED
ENCODING_TYPE                 = "JP2"
ENCODING_TYPE_VERSION_NAME    = version of the JPEG 2000 specification
                                consistent with the data product
INTERCHANGE_FORMAT            = BINARY
UNCOMPRESSED_FILE_NAME        = name of the decompressed file
REQUIRED_STORAGE_BYTES        = approximate total number of bytes in the
                                decompressed data file
DESCRIPTION                   = brief description of the JPEG 2000
                                format, including a reference to the
                                full specification
```

Typically, the DESCRIPTION is given as a pointer to a file called “JP2INFO.TXT” found in the DOCUMENT directory on the same volume.

The subsequent UNCOMPRESSED_FILE object contains a complete description of the data file obtained by decompressing the JPEG 2000 file.

I.3.4 Label Example

The following combined detached label describes a hypothetical JP2 formatted image and the decompressed PDS formatted image derived from it:

```
PDS_VERSION_ID                = PDS3
```

```
/* IDENTIFICATION DATA ELEMENTS */

MISSION_NAME                      = "MARS RECONNAISSANCE ORBITER"
INSTRUMENT_HOST_NAME              = "MARS RECONNAISSANCE ORBITER"
INSTRUMENT_NAME                   = "HIGH RESOLUTION IMAGING SCIENCE
                                     EXPERIMENT"
TARGET_NAME                       = "MOON"
DATA_SET_ID                       = "MRO-L-HIRISE-5-DIM-V1.0"
PRODUCT_ID                        = "CRU_000004_1200_RED2_2"
START_TIME                        = 2005-09-08T23:16:44.863
STOP_TIME                         = 2005-09-08T23:16:51.569
SPACECRAFT_CLOCK_START_COUNT      = 810688604:56542
SPACECRAFT_CLOCK_STOP_COUNT       = 810688611:37300
PRODUCT_CREATION_TIME             = 2005-09-09T15:35:45
(etc.)

/* DESCRIPTIVE DATA ELEMENTS */

(etc.)


OBJECT                            = COMPRESSED_FILE
  FILE_NAME                       = "FILENAME.JP2"
  RECORD_TYPE                     = UNDEFINED
  ENCODING_TYPE                   = "JP2"
  ENCODING_TYPE_VERSION_NAME      = "ISO/IEC15444-1:2004"
  INTERCHANGE_FORMAT              = BINARY
  UNCOMPRESSED_FILE_NAME          = "FILENAME.IMG"
  REQUIRED_STORAGE_BYTES          = 2400000000
  ^DESCRIPTION                    = "JP2INFO.TXT"
END_OBJECT                        = COMPRESSED_FILE

OBJECT                            = UNCOMPRESSED_FILE
  FILE_NAME                       = "FILENAME.IMG"
  RECORD_TYPE                     = FIXED_LENGTH
  RECORD_BYTES                    = 40000
  FILE_RECORDS                    = 60000

/* POINTERS TO DATA OBJECTS */

  ^IMAGE                          = "FILENAME.IMG"

/* DATA OBJECT DEFINITIONS */

  OBJECT                          = IMAGE
    LINES                         = 60000
    LINE_SAMPLES                  = 20000
    SAMPLE_TYPE                   = UNSIGNED_INTEGER
    SAMPLE_BITS                   = 16
    SAMPLE_BIT_MASK               = 2#0011111111111111#
    (etc.)
  END_OBJECT                      = IMAGE
END_OBJECT                        = UNCOMPRESSED_FILE
END
```

I.4 PREVIOUS PIXEL

TBD

I.4.1 PDS Implementation Rules

TBD

I.4.2 Labeling

TBD

I.4.3 Label Example

TBD

I.5 RUN LENGTH

TBD

I.5.1 PDS Implementation Rules

TBD

I.5.2 Labeling

TBD

I.5.3 Label Example

TBD

I.6 ZIP

The Zip method was chosen because the algorithm and supporting software for all major platforms are available without charge to the general user community. The *Info-Zip Consortium* and Info-Zip working group, for example, provide information and software at this URL:

http://www.info-zip.org

This same information is available on line from PDS at:

http://pds.jpl.nasa.gov

I.6.1 PDS Implementation Rules

A volume containing zip files with combined-detached labels as presented below conforms to all established PDS standards *provided both the zip file and its constituent data files are archived*. The unique feature of a Zip-compressed PDS archive volume is that only the zip files appear; the UNCOMPRESSED_FILE objects described by the labels are not present on the volume, but can be obtained by unzipping the zip files provided.

In the interests of long-term archiving, a PDS archive zip file must include all the support files required to completely reconstitute the labeled data files. Specifically, the zipped archive must include not only the data files, but also the label file(s) for the uncompressed data. Ideally, any .FMT files referenced by ^STRUCTURE keywords in the labels should also be included in the zip file.

Note: These additional .LBL and .FMT files do not need to be described by UNCOMPRESSED_FILE objects in the label, because PDS label and format files never require labels. Furthermore, the sizes of these files do not need to be included in the value of the REQUIRED_STORAGE_BYTES keyword. However, the names of these files do need to be included in the list of UNCOMPRESSED_FILE_NAME values.

I.6.2 Labeling

When archiving data in Zip format, two files need to be considered: (1) the zip file itself, and (2) the data file produced by decompressing the zip file. PDS strongly recommends that these two files have the same name but different extensions: ".ZIP" for the zip file and a more descriptive extension (e.g., ".DAT" or ".IMG") for the unzipped file. The ".ZIP" file extension is reserved exclusively for zip-compressed files within the PDS.

PDS does not recommend the practice of compressing multiple data files into a single zip file, unless those files reside in the same directory and have the same name, but different extensions.

For example, if file "ABC.IMG" contains an image and file "ABC.TAB" contains a table of additional information relevant to that image, then both files can be archived in the file "ABC.ZIP". This will minimize the potential confusion for a user who may not be able to locate a desired file because it is hidden inside a zip file with a different name.

Like all PDS data files, both the zipped and the unzipped data files require labels. Both files must be described by a single, detached PDS label file using the combined-detached label approach (see Section 5.2.2). Attached labels are not permitted for Zip-compressed data, because the user must be able to examine the label before deciding whether or not to decompress the file. In a combined-detached label, each individual file is described as a FILE object. Here is the general framework:

```
PDS_VERSION_ID          = PDS3
DATA_SET_ID             = ...
PRODUCT_ID              = ...
   (other parameters relevant to both Zipped and Unzipped files)

OBJECT                  = COMPRESSED_FILE
   (parameters describing the compressed file)
END_OBJECT              = COMPRESSED_FILE

OBJECT                  = UNCOMPRESSED_FILE
   (parameters describing the first uncompressed file)
END_OBJECT              = UNCOMPRESSED_FILE

OBJECT                  = UNCOMPRESSED_FILE
   (parameters describing a second uncompressed file, if present)
END_OBJECT              = UNCOMPRESSED_FILE
END
```

The first FILE object, the COMPRESSED_FILE, refers to the zipped file; additional FILE objects, called UNCOMPRESSED_FILEs, refer to the decompressed data file(s) that the user will obtain by unzipping the first.

The zip file is described via a "minimal label" (see Section 5.2.3). The following keywords are required:

```
FILE_NAME               = name of the zipfile
RECORD_TYPE             = UNDEFINED
ENCODING_TYPE           = ZIP
INTERCHANGE_FORMAT      = BINARY
UNCOMPRESSED_FILE_NAME  = a list of the names of all the files archived
                           in the zipfile
REQUIRED_STORAGE_BYTES  = approximate total number of bytes in the data
                           files
DESCRIPTION             = a brief description of the zipfile format
```

Typically, the DESCRIPTION is given as a pointer to a file called "ZIPINFO.TXT" found in the DOCUMENT directory on the same volume.

The subsequent UNCOMPRESSED_FILE object(s) contain complete descriptions of the data files obtained by unzipping the zip file.

I.6.3 Label Example

The following is an example of a PDS label for a Zip-compressed data file.

```
PDS_VERSION_ID             = PDS3
DATA_SET_ID                = "HST-S-WFPC2-4-RPX-V1.0"
SOURCE_FILE_NAME           = "U2ON0101T.SHF"
PRODUCT_TYPE               = OBSERVATION_HEADER
PRODUCT_CREATION_TIME      = 1998-01-31T12:00:00

OBJECT                     = COMPRESSED_FILE
  FILE_NAME                = "0101_SHF.ZIP"
  RECORD_TYPE              = UNDEFINED
  ENCODING_TYPE            = ZIP
  INTERCHANGE_FORMAT       = BINARY
  UNCOMPRESSED_FILE_NAME   = {"0101_SHF.DAT", "0101_SHF.LBL"}
  REQUIRED_STORAGE_BYTES   = 34560
  ^DESCRIPTION             = "ZIPINFO.TXT"
END_OBJECT                 = COMPRESSED_FILE

OBJECT                     = UNCOMPRESSED_FILE
  FILE_NAME                = "0101_SHF.DAT"
  RECORD_TYPE              = FIXED_LENGTH
  RECORD_BYTES             = 2880
  FILE_RECORDS             = 12
  ^FITS_HEADER             = ("0101_SHF.DAT",     1 <BYTES>)
  ^HEADER_TABLE            = ("0101_SHF.DAT", 25921 <BYTES>)

  OBJECT                   = FITS_HEADER
    HEADER_TYPE            = FITS
    INTERCHANGE_FORMAT     = ASCII
    RECORDS                = 7
    BYTES                  = 20160
    ^DESCRIPTION           = "FITS.TXT"
  END_OBJECT               = FITS_HEADER

  OBJECT                   = HEADER_TABLE
    NAME                   = HEADER_PACKET
    INTERCHANGE_FORMAT     = BINARY
    ROWS                   = 965
    COLUMNS                = 1

    ROW_BYTES              = 2
    DESCRIPTION            = "This is the HST standard header packet
                           containing observation parameters.  It is
                           stored as a sequence of 965 two-byte
                           integers.  For more detailed information,
                           contact Space Telescope Science Institute."

    OBJECT                 = COLUMN
      NAME                 = PACKET_VALUES
      DATA_TYPE            = MSB_INTEGER
```

```
      START_BYTE            = 1
      BYTES                 = 2
    END_OBJECT              = COLUMN
  END_OBJECT                = HEADER_TABLE

END_OBJECT                  = UNCOMPRESSED_FILE
END
```

I.6.4 ZIPINFO.TXT Example

While the ZIPINFO.TXT file is not required, it is strongly recommended that this file be included as part of the process of documenting the contents of a zip file. The following is an example ZIPINFO.TXT file and the type of information that should be included in the ZIPINFO.TXT file:

```
PDS_VERSION_ID              = PDS3
RECORD_TYPE                 = STREAM

OBJECT                      = TEXT
  PUBLICATION_DATE          = 1999-07-26
  NOTE                      = "This file provides an overview of the ZIP
                            file format."

END_OBJECT                  = TEXT
END

Many of the files in this data set are compressed using Zip format.
They are all indicated by the extension ".ZIP".  ZIP is a utility that
compresses files and also allows for multiple files to be stored in a
single Zip archive.  You will need the UNZIP utility to extract the
files.

The SOFTWARE directory on this volume contains a complete description
of the Zip file format and also the complete source code for the UNZIP
utility.  The file format and file decompression algorithms are
described in the file SOFTWARE/APPNOTE.TXT.

It is far simpler to obtain a pre-built binary of the UNZIP application
for your platform.  Binaries for most platforms are available from the
Info-ZIP web site, currently at this URL:

            http://www.info-zip.org/

The same information can also be found a the PDS Engineering Node's web
site, currently at:

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

(This page intentionally left blank.)

PDS Standards Reference
INDEX

A

AAREADME.TXT 9-2, 19-2, A-139, G-2
contents D-2
example D-3
Acronym list
Standards Reference F-1
ALIAS object
definition A-3
aliases
for data types 3-1
ancillary data product
contents 5-15, 6-2
contents (diagram) 6-1
ancillary files 9-4, 19-2, 19-10
ancillary volume 19-2
ARRAY object 13-2, A-1, A-15, A-36
definition A-4
ASCII files
containing markup 9-4
format 9-2
record format 15-1
ASCII tables 10-5, A-119
ASCII_COMPLEX 3-3
ASCII_INTEGER 3-3
ASCII_REAL 3-3
AXIS_ITEMS A-4
AXIS_NAME A-4, A-78, A-80, A-92, A-95

B

BACKPLANE A-77, A-90, H-4
BAND_BIN A-78, A-94
group definition H-3
BAND_STORAGE_TYPE A-65
BAND_SUFFIX A-80, A-94
group definition H-4
BANDS 2-7, A-65, A-79, A-92, A-107, H-3
Bernoulli Disks
delivery medium 11-1
binary data

bit string format 3-5
integer formats 3-5
binary storage formats C-1
binary tables A-122
spare bytes A-122
BINHEX utility 11-3
bit field representation 3-5
BIT_COLUMN object 3-1, A-17, A-122
definition A-8
BIT_DATA_TYPE
standard values 3-1
BIT_ELEMENT object 13-2, A-1, A-5
definition A-11
BIT_STRING 3-3, 3-5
body coordinates
prime meridians 2-3
spin axes 2-3
BOOLEAN 3-3
BOTTOMPLANE A-90, H-3
BYTES A-16, A-55, A-119

C

CALIB subdirectory 8-1, 19-11
calibration data 6-2, 8-1
calibration files 19-11
calibration subdirectory 8-1, 19-11
CALINFO.TXT 19-11
CATALOG directory 19-8, B-1
catalog information 6-2
catalog object files 19-8, B-1
CATALOG object
definition A-12
catalog objects B-1
DATA_SET B-4, B-18, B-19, B27, B-28, B-29
DATA_SET_COLL_ASSOC_DATA_SETS B-11
DATA_SET_COLLECTION B-13
DATA_SET_COLLECTION_INFO B-16
DATA_SET_COLLECTION_REF_INFO B-12
DATA_SET_HOST B-4, B-5, B-18
DATA_SET_INFORMATION B-4, B-19
DATA_SET_MAP_PROJECTION B-23, B-27
DATA_SET_MAP_PROJECTION_INFO B-26
DATA_SET_TARGET B-29
DS_MAP_PROJECTION_REF_INFO B-30
how to supply 19-8

IMAGE_MAP_PROJECTION B-31
INSTRUMENT B-36, B-48
INSTRUMENT_HOST B-41, B-43, B-44
INSTRUMENT_HOST_INFORMATION B-41, B-43
INSTRUMENT_HOST_REFERENCE_INFO B-41, B-48
INSTRUMENT_INFORMATION B-36, B-45
INSTRUMENT_REFERENCE_INFO B-36, B-48
INVENTORY B-49, B-51
INVENTORY_DATA_SET_INFO B-51, B-52
INVENTORY_NODE_MEDIA_INFO B-52
MISSION B-53, B-59, B-60, B-62
MISSION_HOST B-53, B-59, B-63
MISSION_INFORMATION B-53, B-60
MISSION_REFERENCE_INFORMATION B-53, B-62
MISSION_TARGET B-63
PERSONNEL B-64, B-66, B-67
PERSONNEL_ELECTRONIC_MAIL B-64, B-66
PERSONNEL_INFORMATION B-64, B-67
REFERENCE B-28, B-68
SOFTWARE B-75, B-77, B-78, B-79
SOFTWARE_INFORMATION B-75, B-77
SOFTWARE_ONLINE B-75, B-78
SOFTWARE_PURPOSE B-75, B-79
TARGET B-80, B-82, B-83
TARGET_INFORMATION B-80, B-82
TARGET_REFERENCE_INFORMATION B-83
catalog pointer 19-9
CATALOG subdirectory 19-8, 19-12
CATALOG.CAT 19-8
CATINFO.TXT 19-8
CD-Recordable
delivery medium 11-1
CD-ROM
delivery medium 11-1
formatting recommendations 11-1
premastering 11-2
CHARACTER 3-3
CHECKSUM A-67
citations B-68
contents
articles B-70
books B-70
electronic journal articles B-72
electronic publications B-71
PDS data sets B-71
physical media B-72

CITATION_DESC B-21
  example B-5
  formation rule B-21
CLEM-JPEG
  as data compression format I-3
CODMAC numbers. .... *See* data processing level
COLLECTION object A-1, A-4, A-6, A-36
  definition A-15
COLUMN object A-3, A-8, A-85, A-123
  and CONTAINER A-20
  definition A-16
  vectors A-16
combined-detached labels
  and compressed data I-8, I-13
COMPLEX 3-4
Compliance waivers 1-1
COMPRESSED_FILE I-13
compression I-1
compression formats
  CLEM-JPEG I-3
  HUFFMAN FIRST DIFFERENCE I-4
  JPEG 2000 .... I-5
  PREVIOUS PIXEL I-10
  RUN LENGTH I-11
  ZIP I-12
CONFIDENCE_LEVEL_NOTE
  headings B-21
CONTAINER object A-16
  definition A-20
  in TABLE A-139
CUMINDEX.LBL 19-11
CUMINDEX.TAB 19-11
cumulative index 19-11

D

DAT tape
  delivery medium 11-1
data compression I-1
data delivery
  media 11-1
data dictionary files 6-2, 19-12
  PDSDD.FUL 19-12
  PDSDD.IDX 19-12
  See Planetary Science Data Dictionary 19-12
data elements

data identification 5-14
descriptive 5-16
file characteristics 5-10
locally defined 5-18
proposing new 5-15
required and optional 5-11
standards identifiers 5-9
syntax
summary 5-17
data files
contents 19-10
record format 15-1
data identification data elements 5-14
data identification elements
required for ancillary data 5-15
required for Earth-based data 5-15
required for spacecraft data 5-15
data objects A-1
ALIAS A-3
ARRAY A-4
BIT_COLUMN A-8
BIT_ELEMENT A-11
CATALOG A-12
COLLECTION A-15
COLUMN A-16
CONTAINER A-20
DATA_PRODUCER A-27
DATA_SUPPLIER A-29
definition of 5-11
DIRECTORY A-31
DOCUMENT A-33
ELEMENT A-36
FIELD A-38
FILE A-41
GAZETTEER_TABLE A-45
HEADER A-55
HISTOGRAM A-57
HISTORY A-60
IMAGE A-64, A-74
INDEX_TABLE A-69
object definitions 5-16
PALETTE A-74
QUBE A-77
SERIES A-85
SPECTRAL_QUBE A-90
SPECTRUM A-112

SPICE_KERNEL A-115
SPREADSHEET A-118
standard data objects 5-16
TABLE A-123
TEXT A-144
VOLUME A-146
data pointers 14-1
Data Preparation Workbook 1-1
data processing level 6-7
CODMAC numbers 6-6
DATA_PRODUCER object A-29
definition A-27
data product
and PRODUCT_ID 4-1
definition 4-1
file configurations 4-2
labels 5-1
label example 4-2
primary data object 4-1
relation to data set 6-1
relation to data set collection 6-1
seconday data object 4-1
data representation
internal C-1
data set
contents 6-1
definition 6-1
naming and identification 6-2
Non-compliant 1-1
processing level 6-6
relation to data products 6-1
reprocessed, version number 6-8
data set collection
contents 6-1
contents (diagram) 6-1
definition 6-1
naming and identification 6-3
relation to data products 6-1
reprocessed, version number 6-8
data set description
acronyms 6-7
data set type
acronyms 6-7
DATA_SET object A-41, B-18, B-19, B-28, B-29
definition B-4
DATA_SET_COLL_ASSOC_DATA_SETS object

definition ... B-11
DATA_SET_COLLECTION object ... B-11, B-12, B-16, B-17
definition ... B-13
DATA_SET_COLLECTION_DESC ... B-16
DATA_SET_COLLECTION_ID
constituent components ... 6-3
data set type ... 6-7
description ... 6-7
example ... 6-9
standard acronyms and abbreviations ... 6-4
syntax ... 6-4
version number ... 6-8
DATA_SET_COLLECTION_INFO object
definition ... B-16
headings ... B-17
DATA_SET_COLLECTION_NAME
constituent components ... 6-3
example ... 6-9
DATA_SET_COLL_REF_INFO object
definition ... B-12
DATA_SET_DESC ... B-1
headings ... B-20
subheading formats ... B-1
DATA_SET_HOST object
definition ... B-18
DATA_SET_ID ... A-150, A-151, B-52
constituent components ... 6-2
data set type ... 6-7
description ... 6-7
example ... 6-8
satellite and ring names in ... 6-5
standard acronyms and abbreviations ... 6-4
syntax ... 6-4
version number ... 6-8
DATA_SET_INFORMATION object
definition ... B-19
DATA_SET_MAP_PROJECTION object ... B26, B-31
definition ... B-23
DATA_SET_MAP_PROJECTION_INFO object ... B-30
definition ... B-26
DATA_SET_NAME
consituent components ... 6-2
example ... 6-8
satellite and ring names in ... 6-5
DATA_SET_REFERENCE_INFORMATION object ... B-4, B-5
definition ... B-28

DATA_SET_TARGET object B-4, B-5
definition B-29
DATA_SET_TERSE_DESC
example B-5
DATA subdirectory 19-9
data submission 11-1
DATA_SUPPLIER object A-27
definition A-29
data type
data elements 3-1
DATA_TYPE
standard values 3-1
data types
table of data elements 3-2
table of standard values 3-4
DATE 3-4
date format
conventional 7-2
native 7-2
precision 7-1
syntax 7-1
DD_VERSION_ID 5-10
delivery media 11-1
description pointers 14-2
descriptive data elements 5-16
directories
path names 8-4
reserved names 8-1, 10-3
standard directories 8-1
DIRECTORY 8-4
directory names
and ISO 9660 8-4
syntax 8-2
directory naming 8-1
DIRECTORY object A-147
definition A-31
directory paths
and ISO 9660 8-4
syntax 8-4
directory structure
example 8-3
on sequential media 8-4
Distributed Inventory System
DIS.LBL G-2
DOCINFO.TXT 9-2, 19-12, A-144
DOCUMENT 9-2

ASCII version....A-33
DOCUMENT object....9-2
definition....A-33
DOCUMENT subdirectory....19-12, I-8, I-13
documentation....6-2, 9-1
and DOCUMENT object....A-33
ASCII file format....9-3
ASCII version....A-33
criteria for inclusion....9-1
example
attached TEXT....9-5
detached....9-5
with graphics....9-6
file labeling
DOCUMENT object, use of....A-33
format....9-1
HTML....9-1, 9-4
labels for....9-2
markup files....9-4
non-ASCII files....9-5
required ASCII format....9-1
TeX/LaTeX....9-1
validation....9-5
DS_MAP_PROJECTION_REF_INFO object
definition....B-30
DVD media
archive format....11-2
DVD-R
delivery medium....11-1
DVD-ROM
delivery medium....11-1
formatting recommendations....11-2
premastering....11-2
UDF....11-2

E

EARTH_BASE_ID....B-37
EBCDIC_CHARACTER....3-4
ELEMENT object....A-1, A-6
definition....A-36
ENCODING_TYPE....9-5
END statement....5-17, 16-5
Ephemeris time (ET)....7-4
ERRATA.TXT....19-8
Exabyte tape

delivery medium 11-1
extensions
table of 9-3
Extended Attribute Records (XARs)
on delivery disks 11-1
EXTRAS subdirectory 19-13
EXTRINFO.TXT 19-13

F

field delimiters A-142
FIELD object
definition A-38
example A-39, A-40
in SPREADSHEET A-38
file characteristics data elements 5-10
file extensions
reserved extensions 10-3
FILE object A-31, I-8, I-13
definition A-41
explicit 5-8
implicit 5-8, A-41
table of required and optional elements A-42
FILE_NAME 5-11, A-41
file names 10-1
27.3 convention 10-2
8.3 convention 10-2
ISO 9660 Level 1 10-2
ISO 9660 Level 2 10-2
NAIF conventions E-13
reserved extensions 10-3
reserved names 10-3
sequential file names 10-5
syntax 10-2
FILE_RECORDS 5-10
file specification
definition 10-1
example 10-1
file specification and naming
required file examples D-12
FILE_STATE A-78
FLOAT 3-4
floating point 3-6
floating point representation 3-6
FORMAT70 program B-2
format specifications 3-7

for ASCII data files 3-7
for binary data files....3-7

G

gazetteer data 6-2
GAZETTER subdirectory 19-14
gazetteer table 19-14
GAZETTEER_TABLE object
definition A-45
GAZETTER.LBL 19-14
GAZETTER.TAB 19-14
GAZETTER.TXT 19-14
GAZINFO.TXT 19-14
geometry data 6-2
GEOMETRY subdirectory 19-14
GEOMINFO.TXT 19-14
GROUP 12-16
in HISTORY object A-60
in QUBE A-78
PDS use 12-17
group definitions
URL H-1
groups 12-16, H-1
BAND_BIN H-3
BAND_SUFFIX H-4
generic 13-3
example 13-3, 13-5
implementation 13-4
LINE_SUFFIX H-5
PARAMETERS....H-6, H-8
SAMPLE_SUFFIX H-7
specific....13-3
standard groups....13-1
using 13-3

H

HEADER object A-132
definition A-55
secondary data object 4-1
HISTOGRAM object A-78
definition A-57
secondary data object 4-1
HISTORY object
and QUBE A-78

definition A-60
HUFFMAN FIRST DIFFERENCE
as data compression format I-4

I

IBM_COMPLEX 3-4
IBM_INTEGER 3-4
IBM_REAL 3-4
IBM_UNSIGNED_INTEGER 3-4
IEEE_COMPLEX 3-4
storage format C-13
IEEE_REAL 3-2, 3-4
storage format C-10
IMAGE object A-74, A-78
and PALETTE A-74
compression I-1
definition A-64
primary data object 4-1
stored with TABLE object A-138
WINDOW sub-object ....
IMAGE_MAP_PROJECTION object A-78, B-23
definition B-31
include files 19-14
include pointers 14-1
index files 6-2
INDEX subdirectory 19-10
INDEX_TABLE 19-10
contents A-69
INDEX_TABLE object
definition A-69
INDEX_TYPE A-69
INDEX.LBL 19-10
INDEX.TAB 19-10
INDEX.TXT G-2
INDXINFO.TXT 19-10
example D-8
INSTRUMENT object
definition B-37
INSTRUMENT_DESC B-1
headings B-46
subheading formats B-1
INSTRUMENT_HOST object
definition B-41
INSTRUMENT_HOST_DESC,
headings B-43

INSTRUMENT_HOST_ID B-37
INSTRUMENT_HOST_INFORMATION object
  definition B-43
INSTRUMENT_HOST_REFERENCE_INFO object
  definition B-45
INSTRUMENT_INFORMATION object
  definition B-46
INSTRUMENT_REFERENCE_INFO object
  definition B-49
INTEGER 3-4
integer representations
  least significant byte first (LSB) 3-6
  most significant byte first (MSB) 3-6
  signed vs. unsigned 3-6
Integrated Software for Imagers and Spectrometers (ISIS)
  QUBE object A-77
  SPECTRAL_QUBE compatibility A-107
INTERCHANGE_FORMAT A-123
INVENTORY object
  definition B-50
INVENTORY_DATA_SET_INFO object
  definition B-52
INVENTORY_NODE_MEDIA_INFO object
  definition B-53
ISIS .... *See* Integrated Software for Imagers and Spectrometers (ISIS)
ISO 9660
  Level 1 file names 10-2
  Level 2 file names 10-2
ITEM_BITS A-8
ITEM_BYTES 3-1, A-16
ITEM_OFFSET A-8, A-16
ITEM_TYPE (*obsolete*) 3-1
ITEMS 3-1, A-8, A-16

J

Jaz disks
  delivery medium 11-1
JP2INFO.TXT I-8
JPEG 2000
  as data compression format I-5
  example I-8
  file extension I-7
  labeling I-7

K

KERNEL_TYPE
- table of file extensions....A-115

L

label files
- contents....19-10

LABEL_RECORDS....5-10
LABEL_REVISION_NOTE....5-10
LABEL subdirectory....19-14
labeling methods....5-1
- attached....5-1
- case....5-3
- combined detached....5-1
- detached....5-1
- example....5-1
- line lengths....5-1, 5-3

labels....5-1
- and SFDU labels....5-10, 5-17
- attached....5-1
- case....5-3
- character set....5-3
- combined detached....5-1, 5-4, 5-5
- descriptive text pointers....5-16
- detached....5-1
- END statement....5-17
- FILE object
  - explicit....5-8
  - implicit....5-8
- format....5-1, 5-3
  - character set....5-3
- line length....5-1, 5-3
- minimal....5-6, 5-8
- Object Description Language (ODL)....5-1
- object pointers....5-11
  - attached label examples....5-12
  - detached label examples....5-13
- padding....5-3
- pointers
  - to data objects....5-11
  - to descriptive text....5-16
  - to structure files....5-16
- standard data objects....5-16
- standards identifiers....5-9

structure
attached and detached....5-4
attached/detached example....5-5
combined detached.... 5-4, 5-5
combined detached example 5-7, 5-8
minimal....5-8
minimal example....5-9
structure pointers 5-16
LABINFO.TXT....19-14
leap seconds....7-3
representation of....7-3
least significant byte first (LSB) storage format C-6, C-8, C-25
line terminators and delimiters
vis-a-vis byte counts in objects
exclusion of line terminators and delimiters in objects....A-16
LINE_DISPLAY_DIRECTION A-65
LINE_FIRST_PIXEL....B-31
LINE_LAST_PIXEL....B-31
LINE_PREFIX_BYTES A-64
LINE_SAMPLES....A-64
LINE_SUFFIX
group definition H-5
LINE_SUFFIX_BYTES A-64
LINES....A-64
local data dictionaries....5-18
locally defined data elements 5-18
control authority 5-20
custodians 5-20
data dictionary files 19-13
examples....5-18
identification of 5-20
justification for 5-18
namespace 5-20, 12-5
review and use....5-20
scope of use 5-18
local time 7-4
LOCAL_TIME....7-4
logical volumes
multiple logical volumes (definition)....19-1
naming....19-19
single logical volume (definition) 19-1
VOLDESC.CAT....D-13
LOGICAL_VOLUME_PATH_NAME A-150, A-151
LOGICAL_VOLUMES A-150, A-151
LSB integers.... *See* integer representations
LSB_BIT_STRING 3-4

storage format C-25
LSB_INTEGER 3-4
storage format C-6
LSB_UNSIGNED_INTEGER 3-4
storage format C-8

M

MAC_COMPLEX 3-4
MAC_INTEGER 3-4
MAC_REAL 3-1, 3-4
MAC_UNSIGNED_INTEGER 3-5
Management Council 1-1
MAP_PROJECTION_DESC
headings B-26
map resolution 2-5
MAXIMUM_SAMPLING_PARAMETER A-85
MEDIUM_FORMAT A-147
MEDIUM_TYPE A-147
midnight 7-4
representation of 7-4
minimal labels 5-6, 5-8, 5-14, G-2
and compressed data I-8, I-13
MINIMUM_SAMPLING_PARAMETER A-85
MISSION object B-59, B-60, B-62, B-63
definition B-54
MISSION_DESC
headings B-61
MISSION_HOST object
definition B-59
MISSION_INFORMATION object
definition B-60
MISSION_OBJECTIVES_SUMMARY
headings B-61
MISSION_REFERENCE_INFORMATION object
definition B-62
MISSION_TARGET object
definition B-63
most significant byte (MSB) first integers C-2, C-4, C-23
MSB integers .... *See* integer representations
MSB_BIT_STRING 3-5
storage format C-23
MSB_INTEGER 3-5
storage format C-2
MSB_UNSIGNED_INTEGER 3-5
storage format C-4

N

N/A constant 17-1
NAIF
  file naming conventions E-13
NAIF Toolkit A-115
  directory structure E-1
  using E-12
NAME A-41
namespace .... *See* locally defined data elements
NASA processing levels. .... *See* data processing level
native date 7-2
native time 7-3
  examples 7-3
Not Applicable constant 17-1
NULL constant 17-1

O

OBJECT 12-15
object definitions 1-3
  format 5-16
  URL A-1
Object Description Language (ODL) 5-1
  character set 12-3
  comments 12-13
  date and time formats 12-8
  date formats 12-9
  END statement 12-13
  file format 12-13
  identifiers
    reserved 12-12
    syntax 12-11
  implementation
    date and time 12-9
  implementation notes 12-2
  integer formats 12-6, 12-7
  language summary 12-22
  lexical elements 12-6
  numeric values 12-18
  Parameter Value Language (PVL) 12-26
  PDS implementation 12-2
    date and time 12-9
    sets 12-22
    symbolic literals 12-21
  PVL guidelines 12-27

PVL restrictions on archive files 12-26
real number formats 12-7
revision notes 12-24
version 0 12-25
version 1 12-24
sample data label 12-1
sequences 12-21
sets 12-22
special characters 12-12
specification 12-1
statements 12-12
assignment 12-14
GROUP 12-16
OBJECT 12-15
pointer 12-15
symbol strings 12-11
symbolic literals 12-20
text string values 12-19
text strings 12-11
time formats 12-9
units of measure 12-18
object pointers 5-11
attached label examples 5-12
formats 5-11
syntax 5-13
objects .... *See* catalog objects, data objects
generic 13-1
example 13-1
primitive 13-2
specific 13-1
example 13-1
standard objects 13-1
objects, data
SERIES
use of spare fields A-127
SPECTRUM
use of spare fields A-127
observing campaign
as MISSION B-54

P

PALETTE object A-78
definition A-74
secondary data object 4-1
Parameter Value Language (PVL) 12-2, 12-26

PARAMETERS group
alias H-6, H-8
definition H-6, H-8
PC_COMPLEX 3-5
storage format C-17
PC_INTEGER 3-5
PC_REAL 3-5
storage format C-14
PC_UNSIGNED_INTEGER 3-5
PDS catalog
on-line search A-1, H-1
references on-line B-70
PDS groups .... *See* data groups
PDS objects .... *See* catalog objects, data objects
PDS Software Inventory .... *See* time standard
PDS time standard B-75
PDS_USER_ID B-64
PDS_VERSION_ID 5-9
PERSONNEL object B-66, B-67
definition B-64
PERSONNEL_ELECTRONIC_MAIL object
definition B-66
PERSONNEL_INFORMATION object
definition B-67
physical media formats 11-1
physical media
organization 19-1
Planetary Science Data Dictionary (PSDD) 1-1, 3-1, 5-1, 5-14, 5-16
file labeling 19-13
identifying the version in a label 5-10
local data dictionaries 5-18
locally defined data elements 5-18, 19-13
PDSDD.FUL 19-12
PDSDD.IDX 19-12
pointers
catalog 19-9
data 14-1
in attached labels 14-1
in detached labels 14-1
description 14-2
include 14-1
rules for resolving 14-3
structure pointers 5-16
to descriptive text 5-16
use in labels 14-1
prefix or suffix data

in QUBE object A-83
in TABLE object A-139, A-143
PREVIOUS PIXEL
as data compression format I-10
primary data object 4-1
IMAGE object 4-1
QUBE object 4-1
SERIES object 4-1
SPECTRUM object 4-1
SPREADSHEET object 4-1
TABLE object 4-1
primitive data objects A-1
primitive objects
ARRAY A-4
BIT_ELEMENT A-11
COLLECTION A-15
ELEMENT A-36
processing level number .... *See* data processing level
PRODUCT_ID 4-1, A-115

Q

QUBE object
and HISTORY object A-78
definition A-77
primary data object 4-1
SPECTRAL_QUBE A-90

R

REAL 3-5
RECORD_BYTES 5-10, A-130
record formats 15-1
blocking 15-1
FIXED_LENGTH 15-1
STREAM 15-2
UNDEFINED 15-3
VARIABLE_LENGTH 15-2
VAX counted byte strings 15-2
RECORD_TYPE 5-10, 15-1
reference coordinates 2-1
body-fixed 2-2, 2-3
data elements 2-1
planetocentric 2-2, 2-3
planetographic 2-2, 2-3
ring systems 2-3

ring systems, data elements 2-3
reference frames
B1950 2-1
standard inertial (J2000) 2-1
REFERENCE object B-4, B-5, B-12, B-30, B-45, B-49, B-62, B-83
definition B-68
reference surface models 2-5
digital image model (DIM) 2-5
digital terrain model (DTM) 2-5
references
citations B-68
contents B-70
on-line access to PDS cataloged B-70
what to reference B-68
REFERENCE_KEY_ID B-5, B-12, B-28, B-30, B-45, B-62, B-83
relative time 7-4
REPETITIONS A-21
required files
examples D-1
REQUIRED_STORAGE_BYTES I-12
ROOT directory files 19-8
rotation direction
of Solar System bodies 2-2
ROTATIONAL_ELEMENT_DESC
headings B-26
ROW_BYTES A-130
ROW_PREFIX_BYTES A-64, A-86
use A-138
ROW_SUFFIX_BYTES A-64
use A-138
RUN LENGTH
as data compression format I-11

S

safing data G-1
criteria G-1
procedures G-1
standards G-1
SAMPLE A-74
SAMPLE_BITS A-64
SAMPLE_DISPLAY_DIRECTION A-65
SAMPLE_FIRST_PIXEL B-31
SAMPLE_LAST_PIXEL B-31
SAMPLE_SUFFIX
group definition H-7

SAMPLE_TYPE A-64
standard values 3-1
sampling parameter data
in SERIES object A-89
SAMPLING_PARAMETER_INTERVAL A-85
SAMPLING_PARAMETER_RESOLUTION 18-1
SAMPLING_PARAMETER_UNIT 18-1
SAVED data G-1
SCALING_FACTOR A-67
sclk .... *See* spacecraft clock count (sclk)
secondary data object 4-1
HEADER object 4-1
HISTOGRAM object 4-1
PALETTE object 4-1
SEQUENCE_NUMBER A-31, A-147
sequential media A-31
SERIES object A-16, A-130
definition A-85
primary data object 4-1
TIME_SERIES A-85
use of spare fields A-143
SIDEPLANE H-7
SOFTINFO.TXT 19-15, A-144
contents D-9
example D-9
software
packaging for delivery 11-2
PDS inventory of tools and libraries B-75
SOFTWARE object B-77, B-78, B-79
definition B-75
software directory structure
NAIF E-1
software files 6-2
SOFTWARE subdirectory 19-15
SOFTWARE_INFORMATION object
definition B-77
SOFTWARE_ONLINE object
definition B-78
SOFTWARE_PURPOSE object
definition B-79
SOURCE_PRODUCT_ID A-115
spacecraft clock count (sclk) 7-3
syntax 7-4
SPACECRAFT_ID B-37
SPACECRAFT_CLOCK_START_COUNT 7-3
SPACECRAFT_CLOCK_STOP_COUNT 7-3

SPARE bytes ...A-127, A-139
spare fields
  use in TABLE, SPECTRUM and SERIES objects ...A-143
SPECTRAL_QUBE object
  BACKPLANE ...H-4
  BOTTOMPLANE ...H-5
  compatibility with ISIS ...A-107
  definition ...A-90
  labeling ...A-97
  SIDEPLANE ...H-7
SPECTRUM object ...A-16, A-130
  definition ...A-112
  primary data object ...4-1
  use of spare fields ...A-143
SPICE kernels
  labeling ...A-115
SPICE system
  kernel file extensions ...A-115
SPICE_KERNEL object
  definition ...A-115
SPREADSHEET object ...
  CSV file format ...A-119
  definition ...A-118
  example ...A-120
  field delimiters ...A-119
  formats ...A-119
  primary data object ...4-1
SRC directory
  in NAIF distribution ...E-4
Standard Formatted Data Unit (SFDU) ...5-10, 5-17
  AMMOS usage ...16-5
  definition ...16-1
  examples
    FIXED_LENGTH file ...16-7
    STREAM file ...16-7
    UNDEFINED file ...16-7
    VARIABLE_LENGTH file ...16-8
  exceptions ...16-8
  I class ...16-2, 16-5
  K class ...16-5
  usage in PDS products ...16-1
  versions ...16-1
  Z class ...16-2, 16-5
standards identifier data elements ...5-9
START_BYTE ...A-4, A-15, A-36, A-124
START_TIME ...7-3, A-85

STOP_TIME 7-3, A-85
storage formats
  binary integers 3-5
STREAM 15-2
STRUCTURE pointer A-128
STUFFIT utility 11-3
SUN_COMPLEX 3-5
SUN_INTEGER 3-5
SUN_REAL 3-5
SUN_UNSIGNED_INTEGER 3-5
Syquest disks
  delivery medium 11-1
Systeme Internationale d'Unites (SI) 18-1

T

TABLE object A-16, A-64, A-78
  and CONTAINER A-139
  ASCII field delimiters A-142
  ASCII tables A-124
  binary tables A-127
  definition A-123
  INDEX_TABLE A-69
  more than one in a single file A-136
  PALETTE A-74
  primary data object 4-1
  SERIES object A-85
  SPARE bytes A-127, A-139
  SPECTRUM object A-112
  stored with IMAGE object A-138
  STRUCTURE pointer A-128
  use of spare fields A-143
  variations A-130
tables
  multiple tables in single file A-136
tape volumes 8-4
tar utility 11-4
target
  acronyms 6-4
  named features 19-14
TARGET object B-82, B-83
  definition B-80
TARGET_INFORMATION object
  definition B-82
TARGET_NAME
  acronym list 6-4

TARGET_REFERENCE_INFORMATION object B-80
  definition B-83
TEXT 9-2
text
  plain text formatting A-144
TEXT object 9-2
  definition A-144
TIME 3-5
time format
  native 7-3
  precision 7-2
  syntax 7-1
time standard 7-3
time tags
    format 2-1
two's complement 3-4, C-2, C-6

U

UNCOMPRESSED_FILE I-8, I-13
UNCOMPRESSED_FILE_NAME I-12
UNDEFINED 15-3
units of measure 18-1
  default units 18-1
  SI prefixes 18-2
  SI units 18-1
  supplementary units 18-3
  symbols 18-1
Universal Time Coordinated (UTC) .... *See* time format
  date/time formats, use in
    UTC, use of 7-2
  in labels and catalog files 5-18
UNK constant 17-1
Unknown constant 17-1
UNSIGNED_INTEGER 3-5

V

VARIABLE_LENGTH 15-2
VAX_BIT_STRING 3-5
VAX_COMPLEX 3-5
  storage format C-22
VAX_DOUBLE 3-5
VAX_INTEGER 3-5
VAX_REAL 3-5
  storage format C-18

VAX_UNSIGNED_INTEGER....3-5
VAXG_COMPLEX....3-5
  storage format....C-22
VAXG_REAL....3-5
  storage format....C-18
version number....6-8
VICAR headers....A-55
VOLDESC.CAT....19-8, 19-18, A-12, A-147
VOLDESC.SFD....19-8
VOLINFO.TXT....19-12, A-144
VOLUME....8-4
volume....19-8
  ancillary volumes....19-2
  definition....19-1
  IDs....19-17
    exceptions....19-19
  logical volume naming....19-19
  names....19-17
volume index....19-11
VOLUME object....A-12, A-27, A-29, A-31, A-41, A-150
  definition....A-146
volume organization and naming....19-1
volume set
  definition....19-1
  IDs....19-18
  names....19-18
  organization
    many data sets, many volumes....19-2, 19-7
    many data sets, one physical volume....19-6
    many data sets, one volume....19-1
    one data set, many volumes....19-1, 19-4
    one data set, one volume....19-1, 19-3
    recommendations....19-7
  types....19-1
VOLUME_FORMAT....A-147
VOLUME_ID....19-18, G-1
VOLUME_SET_ID....19-18
VOLUME_VERSION_ID....19-18
volumes, logical....19-10

W

Waivers (compliance)....1-1
WINDOW object....
WORM disk
  delivery medium....11-1

Z

ZIP
    as data compression format I-12
    compression ratios compared to JPEG 2000 .... I-7
    example I-14
    labeling I-12
Zip disks
    delivery medium 11-1
ZIPINFO.TXT I-13, I-15