DRAFT

Scientific Data Models and Methods

Mike Martin

08-16-2011

Summary

This document describes the data models for several formats for storing scientific data. It is intended to characterize the structural components of these formats and describe the software methods used to access the data. The model descriptions reveal two fundamental approaches, "variable" based and "object" based. For the variable based approaches, the main enhancements have been adding layers to support grouping of variables in a single file and support for more complex variable linkages (grids, swaths). Two of the object-based approaches (HDF4 and PDS3) have been simplified in subsequent versions. The other object-based approach (FITS) began with only three basic objects and has remained very stable. ... to be continued ...

Basic Architectural Features

The Common Data Format (CDF) started out with multidimensional "variables" and metadata "attributes" which can be global or associated with specific variables. This format does not allow a table structure with records. The Network Common Data Format (NetCDF3) added named "dimensions". These are used to relate variables and a dimension of "unLimited" can be used to create a table structure for all the variables that use that dimension name. NetCDF3 was extended to add a "types" class to allow the explicit definition of "compound" structures (tables) , enumerations and ragged arrays. NetCDF4 or the Common Data Model (CDM) encompasses netCDF and Hierarchical Data Format (HDF) version 5 and adds hierarchical "groups", eliminates the "types" class, and renames "compound" to "structure" variables, which can be also be defined as "sequences". HDF4 provides a variety of object types. It includes several types of two dimensional "images" (8 bit, 24 bit and general multi-component), a "palette", a "scientific dataset" which is a multidimensional array, a "vdata" which is a table structure, a "vgroup" for grouping objects and an "annotation" for providing metadata. HDF5 partly eliminated the multitude of objects to focus on "groups", "datasets" (which is a synonym for variables) and global and variable "attributes". It still provides named classes of datasets for images, palettes and tables. Unlike the scalar or one dimensional attributes found in the CDF formats, HDF5 attributes can be multidimensional arrays with some limitations on how they are defined and accessed. HDF-EOS builds on HDF4 or HDF5 with a set of ODL labels which provide additional metadata for point, swath or grid structures. The OPeNDAP Data Access Protocol adds a "Dataset" as a container for groups, variables, and attributes, and adds a "grid" composed of a multidimensional array linked to one-dimensional axis descriptions. FITS includes a multidimensional "image", and both ASCII and binary "table" extensions with slightly different characteristics. A random groups format is no longer actively supported. PDS3 provided a language for describing virtually any data format as well as format specification for a variety of object types. The basic objects included the "file", multidimensional "image", "table" and multidimensional "qube". Both the image and qube objects could include tabular data prefixes or suffixes to allow ancillary data to be embedded with the data array, a common feature of telemetry streams. The table object has several variations called header, spectrum and series and is composed of "columns". There is also a "spreadsheet" with "fields". PDS4 provides four structures, the "array_base", the "table_base", the stream and the encoded byte stream (or external format). Each object must have its own contiguous data space. Numerous classes are built on these bases such as image_2d, table_character, table_binary, etc. All metadata is contained in a separate XML file and there is an elaborate class structure for organizing metadata.

Table 1. Data Model Comparisons

Format	CDF	netCDF3	CDM	HDF5	HDF4	PDS3	PDS4	FITS
Age	30	24	3	10	24	20	0	30
Data Objects	1	1	1	4	6	16	Many	3
Data model components	Variable, Attribute	Variable, Dimension, Attribute	Group, Variable, Dimension, Structure, Sequence, Attribute	Group, Dataset (variable), Attribute, plus image, palette, table and packet table apis	Scientific Dataset (variable), 8, 24, and General Raster Image, Palette, Vdata (table), Vgroup, Annotation	File, Image, Palette, Histogram, Items, Array, Element, Qube, Table, Column, Bit_Column, Spreadsheet, Field, Container, History, Attribute	Array, Axis, Element, Table, Stream, Record, Field, Attribute + ~18 derived classes+ encoded formats	Image, ASCII Table, Binary Table, Field, Attribute
Subclasses of components	NO	NO	NO	YES	YES	YES	YES	NO
Metadata	Embedded or Skeleton	Embedded or CDL	Embedded	Embedded	Embedded	ODL labels	XML labels	Card Images
XML support	CDFML	NcML	NcML	HDFDump option	HDF mapping	NO	YES	NO
Bit Fields	NO	NO	NO	YES	YES	YES	YES	?
MultiFile	OPTIONAL strictly defined	NO	YES	YES	YES	OPTIONAL	REQUIRED	NO
Compression	Yes, strictly defined	YES	YES	YES	YES	YES	NO	YES
Encoding Type	NO	NO	NO	NO	NO	YES	YES	NO
Character table subclass	NO	NO	NO	NO	NO	NO	YES	YES
Reference External Formats	NO	NO	?	?	NO	YES	YES	NO
Tiling/Chunking	?	YES	YES	YES	YES	NO	NO	YES
Mixed Data Objects	NO	NO	NO	NO?	NO	YES	NO	NO
Datatypes	BE and LE char, uchar,	byte, char, short, int,	char, byte, ubyte,	BE and LE ints and	BE and LE char1,uchar	BE and LE ints and	BE and LE int2, 4, 6, 8,	byte, short, long, float,

	byte, int1, 2, 4, uint1, 2, 4, float, double	float, double	short, ushort, int, uint, long, ulong, float, double, string, enumeration, opaque	floats, char, bitfield,string, opaque, enumeration, datetime, reference, variablelength	1, int1, 2, 4, uint1, 2, 4, float4, float8	floats, char, bitfield,string, datetime	16, uint1, 2, 4, 8, 16, float, double, complex8, complex16 +special types?	double, [boolean, bit, complex for binary tables]
Attribute Types	scalar or 1d array of datatype	scalar or 1d array of datatype	scalar or 1d array of datatype	Same as variable, but with some restrictions	Annotation?	Int, float, string, boolean or 1d array	30 different types	Boolean, int, float, complex, string
User Defined Data Types	NO	NO	NO	YES	NO	NO	NO	NO
Platform specific types	YES	NO – XDR	YES	YES	YES NATIVE	YES	YES for INTs	NO IEEE+MSB
UTF-8 support	NO	NC	YES	YES	NO	NO	YES	NO
Schema Classes	13	5	?	42	Many	N/A	44 basic, 72 object, 53 product	N/A
Recognized by	ISTP/IACG	OGC	UNIDATA	ESDIS	ESDIS	PDS	PDS	IAU
Maintained by	SPDF	UNIDATA	UNIDATA	HDFGROUP	HDFGROUP	PDS	PDS	HESARC/NOST
Index Base	0	0	0	0	0	1	0	1
Array axes limit	10	Nominally 7	Nominally 32	Nominally 32	Nominally 32	6	16	999
Magic Number	0xCDF26002	CDF\001	N/A	894844460 d0a1a0a	0E031301	CCSD, NJPL, PDS_	NONE	SIMPLE
File Extensions	CDF, V0, V1, …	NC	N/A	H5, HDF5	HDF, H4, HDF4	LBL, FMT, IMG, IMQ, IBG, TAB, DAT, CUB, QUB, OBJ	XML+???	FIT
Software Library	C,Fortran,Java	C,Fortran,Java	C,Fortran,Java	C, C++, Fortran,Java	C,Fortran,Java, Python	C	NONE	C,Fortran,Java
BUILT-IN IDL	YES	YES	YES	YES	YES	NO	NO	NO
BUILT-IN MATLAB	YES	YES	YES	YES	YES	NO	NO	YES
Utilities	Convert, Compare, Dir, Edit, Export, Inquire, Merge, Stats,	Ncbrowse, Ncview, Ncgen, Ncdump, Nccopy, Ncgen3	Uses netCDF and HDF formats and tools	HDFView,h5dump, h5diff, h5ls, h5check, h5stat, h5repack, h5repart,h5i	HDFView, h4redeploy, hdfcomp, hdfed, hdfls, hdfimport, hdfpack,	Nasaview, label design, label validation, volume validation	N/A	Fv (fitsview), fpack, funpack, topcat, fcopy, fdump,

	Makecdf, DTWS, SkeletonTable SkeletonCDF			mport, h5jam, h5unjam, h5copy, h5mkgrp, h4toh5, h5toh4	hdfunpac, hdiff, hrepack, vmake, vshow			fplot, fverify, etc.
API Classes	CDF (file), Attribute, Entry, Variable	File, Attribute, Dimension, Variable	File, Group, Attribute, Dimension, Sequence, Structure, Variable	File, Group, DataType, DataSpace, Dataset, Property, Attribute	File, Group, Datatype, GRImage, SDS, Vdata	Label, Object, Image, Table, Column, Qube, Spectrum	N/A	Header, ASCII Table, BinaryTable, Column, Image, Undefined

Appendix A. Common Data Format (CDF)

CDF is a data format supported by the Space Physics Data Center at GSFC. It consists of 12 main classes and 8 special attributes. The CDF model consists of *variables* which are made up of *records* which contain multidimensional *elements*. Metadata is provided in global or *variable attributes* which contain one or more metadata *entry*'s. CDF provides a wide range of signed and unsigned data types in a variety of machine-dependent formats. Data can be stored across multiple files, can be organized in row or column major format. Storage can be optimized with record and dimension variances which allow non-varying values to be eliminated. Several types of compression are supported, including Run-Length Encoding, Huffman, Adaptive Huffman and GZIP. Software API's are available in C, Java, Fortran and Perl. About 10 utilities are provided as well as the Data Translation Web Service to comvert to and from CDF format. Routines to access CDF files are provided in IDL and Matlab. The Common Data Format Markup Language (CDFML) provides a mechanism for creating a CDF file from an XML document or for creating an XML document from a CDF file. The following paragraph describes the CDFML architecture.

A CDFML file has an attribute "name" which contains the cdf file name. It has subclasses *cdfFileInfo, cdfGAttributes, cdfVariables*. *FileInfo* has attributes "fileformat" (SINGLE or MULTI), "majority" (ROW or COLUMN) , "encoding" (IBMPC, MAC, etc.), "compression" (RLE, HUFFMAN, etc.) and "negToPosFp0" (ENABLE, DISABLE). The cdfGAttributes (global attributes) subclass has attribute "name" and a subclass *entry*. *Entry* has attributes "entryNum", "cdfDatatype" and a value. The subclass *cdfVariables* has a subclass *variable.* The subclass *variable* has an attribute "name", and has subclasses *cdfVarinfo, cdfVAttributes* and *cdfVarData.* The subclass *cdfVarinfo* has attributes "cdfDatatype", "numElements" (for strings), "dim"(dimensions), "dimSizes", "recVariance" (values change from record to record), "dimVariences" (values change within dimension) and "numRecordsAllocate". The subclass *cdfVAttribute* (variable attributes) contains *attributes* and *entrys,* like the *cdfGAttribute*. The subclass *cdfVarData* contains as many subclass *record* entries as were identified in the "numRecordsAllocate" attribute. Each *record* has attribute "recNum" and contains a list of data values. There is also an *orphanAttributes* subclass.

Appendix B. Network Common Data Formant (NetCDF).

NetCDF is supported by UNIDATA. There are two forms of netCDF, classic (netCDF-3) and enhanced (netCDF-4). Enhanced is downwardly compatible with classic and also supports access to HDF5, GRIB files or OPeNDAP datasets. Software libraries are available in C, C++, Fortran and Java.

NetCDF is very similar to CDF with the main differences being support for named *dimension*'s, all data must be in a single file and only big-endian XDR format datatypes are supported.

A netCDF classic *file* has attributes "id" and "uri". It has subclasses *dimension*, *variable* and global *attribute*. A dimension has attributes "name", "length", and "isUnlimited". Only one *dimension* can be unlimited. A *variable* has attributes "name", "shape" and "type" and subclasses *attribute* and *value*. An *attribute* has attributes "name", "type", "separator" and "value" which is a scalar or one dimensional array of strings or numeric values . The type is implied from the value that is supplied when the *attribute* is defined. The "isUnlimited" attribute allows the construction of records where all variables that use the isUnlimited *dimension* are stored contiguously on a per record basis, thus creating a table or database record structure. The netCDF classic format has a 64-bit variant that allows large variables and files.

The netCDF4 or Common Data Model format uses a constrained HDF5 for data storage. The netCDF4 software can access *files* or *datasets* stored in HDF5, netCDF3, OPeNDAP or several other formats. The *file* or *dataset* contains one or more hierarchical *groups*. A group has attribute "name", which is "/" for the default root group and may contain *attribute*s, *dimension*s and *variable*s as well as *enumtypedef*s and nested *groups*. The *dimension* defines the shape of the *variable* and may be shared between *variable*s to associate them. A *dimension* has attributes "name", "length", "orgName", "isUnlimited", "isShared" and "isVarableLength". A *variable* has attributes "name", "shape" and "type" and subclasses *attribute, values, variable*. An *attribute* has attributes "name", "type", "separator" and "value" which is a scalar or one dimensional array of strings or numeric values. Variables can be grouped into a *structure* with "members" which can be organized as a s*equence*. The netCDF 4 model is different than the enhanced model by using the *types* subclass to allow the definition of *enums*, variable length arrays or *vlens*, and *compound* structures.

Appendix C. Hierarchical Data Format (HDF) and related formats.

HDF was developed at NCSA at UIUC. It is now maintained by the HDFGroup. The original HDF4 data model consists of a directory and a collection of data objects. Eight basic classes are supported, the *Scientific Dataset* (SDS) multidimensional array, several two dimensional raster image types, including an *8-bit raster image*, a *24-bit raster image*, and a general raster image with multi-component pixels, an 8-bit color *palette*, a *table* (Vdata) or sequence of records, an *annotation* stream of text that can be associated with any object and a *group* (Vgroup) structure for grouping objects. Each class has a multitude of attributes. A raster image has attributes image "name", "index", "number of components", "number type", "interlace mode", "dimensions" and "number of attributes". An SDS has "name", "rank", "dimensions", "data type" and "number of attributes". A vdata has a "name", "record count", "interlace mode" (FULL or NONE), and a list of "fields" and a "record size". A vgroup has "name" and "class".

HDF5 was designed to address inadequacies of HDF4 with respect to performance, scale and flexibility. It a uses the term *dataset* to represent a data object or what is called a *variable* in CDF or netCDF. The HD5 data model includes a *file* with a "location" pointing to a url. Every HDF5 file has a root group named "/" which contains *groups, attributes* or *datasets* (variables). A group has the attribute "name" and is a hierarchical container which acts like folders in a file system. A *dataset* (*variable)* has a "name", "shape" and "type" and has subclass *attributes*. *Datatypes* in HDF5 include byte, short, int, long, float, double, String, BitField, Enumeration, Date/Time, Opaque, Reference and VariableLength. A *dataspace* (dimension) defines the shape of a *variable*. An *attribute* is associated with an object and is similar to a dataset but can only be accessed via an object, can only be accessed in

a single I/O operation, should be small and cannot contain sub-attributes. An HDF5 dataset (variable) can have up to 32 dimensions.

Two other formats, HDF-EOS and OPeNDAP use HDF4 or HDF5 for data storage.

HDF-EOS defines a geo-located *point*, *swath* (time-ordered scanlines) or *grid* (rectilinear array). It uses text files in Object Description Language (ODL) to describe these objects. The point is a series of data fields taken at irregular times at scattered locations (e.g.lat, lon, time, temp). The swath consists of data fields, geo-location fields (time or lat/lon), named dimensions, dimension maps (offset and increment relating a data field and a geo-location field) and an optional index (for irregular dimension maps). The grid is similar to a swath, but the geo-location fields are computed from a set of projection equations.

OPeNDAP is a server model that provides access to data in HDF4 and HDF5, netCDF and other formats. It also provides its own data model which is similar to the Common Data Model. A text file called the Dataset Descriptor Structure (DSS) describes the data format and another text file called the Dataset Attribute Structure (DAS) provides metadata. There is also an XML version of the descriptive information called the DDX. The data can also be included in the XML document to produce a DDXData file. OPeNDAP defines a *dataset* as a container for *variables* and *attributes*. OPeNDAP defines four constructor types. The *structure* is a container for either variables or attributes (the *dataset* is a structure). The *array* defines a one-dimensional array (multidimensional arrays are defined as arrays of arrays). The *sequence* is an ordered collection of structures. The *grid* is an association between a data array and a collection of map arrays (lat, lon, time). The OPeNDAP server provides access to the ddx, dds, das as well as info, html and rdf versions of descriptive information. OPeNDAP only supports XDR numeric data types.

Appendix D. Flexible Image Transport System (FITS).

The FITS format began as a transport format for ground-based astronomy images, but has grown into a universal format for astrophysics data files. It is primarily supported by NASA GSFC organizations (HESARC and NOST). A FITS file contains "header units" and "data units". The header unit contains 80 byte ASCII card images of "attributes" that describe the contents of the data unit. A standard FITS file contains a "primary" multidimensional data array with up to 999 axes. The file can also contain "extensions" which include an "image", "table" or "bintable". The image extension is a multidimensional array of pixels like the primary array. The table extension is used for ASCII text tables such as astronomical catalogs. The bintable extension provides a binary table composed of columns that can be uniform or ragged arrays and multidimensional arrays are supported by some software. Each header or data unit is a multiple of 2880 bytes. FITS datatypes include 8 bit char or unsigned integer, 16 and 32 bit signed integers, IEEE 32 and 64 bit floating point.

Appendix E. Planetary Data System PDS3 and PDS4

The PDS3 data descriptive language was developed and is maintained by the Planetary Data System (PDS). PDS3 uses the keyword = value Object Description Language (ODL) to describe data objects and groups and to provide metadata *attributes*. These labels can be included at the front of a data file or in a detached ".lbl" file with a pointer and offset to the data file. An implicit or explicit *file* object contains primary data objects which include a multidimensional *image*, a multidimensional *cube*, a *table* and a *spreadsheet*. The image has *lines*, *samples* and *bands* and can have a tabular *prefix* or *suffix*. The cube has *axes* and can have *backplanes*. The *table* has *rows* and *columns*. A *column* can

have "items" making it an array or can hold "bit_columns". A *table* can have a *container* which holds "repetitions" of *columns*. There are several subclasses of the table object, the *spectrum, series, palette, gazetteer_table*. These subclasses might include special attributes such as sampling_parameters for the series subclass. A *spreadsheet* is a delimited text file with *fields.* There is also a set of minimally defined ancillary objects including the *document, header, histogram, history, spice_kernel* and *palette* and a set of primitive objects, the *array, collection* and *element*. PDS3 is capable of describing practically any data format including complex telemetry streams. It allows several data compression schemes, however these are not well supported by software. The C language label utilities and object access library allow operations on some data objects and a Java library allows operations on labels but not data objects. The NASAVIEW utility allows display and inspection of most but not all PDS3 data objects.

The PDS4 data format is being developed and maintained by the Planetary Data System (PDS). PDS4 uses XML to describe data objects and provide metadata *attributes*. The file_area contains a pointer and offset to the actual data file as well as a description of the objects contained in that file. Four object classes are defined, *array_base, table_base, stream_delimited* and *encoded_byte_stream*. The *array_base* is composed of *array_axis* and *array_element* entries. The *array_base* is further qualified into *array_2d* or *array_3d* sub-classes. The *array_2d* sub-class is further qualified into *array_2d_image, array_2d_spectrum,* and *array_2d_map*. The *array_3d* subclass is further qualified into *array_3d_image, array_3d_spectrum* and *array_3d_movie*. All the sub-classes with the same dimensions are structurally identical. The *table_base* is a fixed length structure composed of *fields*. The *table_base* is further qualified into a *table_character, table_character_grouped, table_binary* and *table_binary_grouped*. Binary tables may contain character fields. The "grouped" sub_class allows the definition of a *sequence* of "repetitions" of fields. The *stream_delimited* class contains *records* and *fields*. The *encoded_byte_stream* is used to describe external data formats such as a *file_pdf* or *encoded_image*. Several other ancillary classes are defined including *XML_schema, header, spice_kernel_text*, and *spice_kernel_binary.* A wide range of BE and LE data types as well as bit fields are supported. Data objects are not allowed to be interleaved. Compression is not allowed on data objects unless it is a feature of an accepted encoded_byte_stream format (e.g. jpeg). Files can be compressed using the zip utility.