

HDF- A Suitable Scientific Data Format for Satellite Data Products

Sk. Sazid Mahammad, Debajyoti Dhar and R. Ramakrishnan

Data Products Software Division

Space Applications Centre, ISRO, Ahmedabad – 380 015

Ph: +91-79-691 4145/ 4752 Email : sazid, deb, rama@ipdpg.gov.in

Key words: hierarchical data format, HDF5, HDF-EOS, spatial data products

Abstract

The space borne remotely sensed images in digital form have gained wide spread popularity over the last decade with the advances in the field of *digital image processing*, *geographical information system (GIS)* and evolution of computer hardware and software. Currently used data formats for raster image data like PNG, GIF, BMP, and TIFF have a common limitation in the data that they can store. In contrast, the *Hierarchical Data Format (HDF)* designed by the US National Centre for Supercomputing Applications (NCSA) is a general purpose scientific data format which can store any datatype. HDF is supported by a freely available function library by the same name and a set of command line utilities and a test suite. HDF is a multi-object file format used for the transfer of graphical and numerical data between machines. It is versatile, flexible, extensible and portable. HDF files are self-describing. The term “self-description” means that, for each HDF data structure in a file, its metadata (comprehensive information about the data and its location in the file) is also available as part of the format. Many types of data can be included within an HDF file. For example, it is possible to store symbolic, numerical and graphical data by using appropriate HDF data structures.

HDF5, the latest version of HDF, has been adopted as one of the digital data product formats for the forthcoming Indian remote sensing satellites (IRS). HDF has also been selected by the Earth Observation System - Data and Information System (EOSDIS) project as the format of choice for distribution of standard products. This version is called HDF-EOS. It is based on HDF with additional data structures to cater to satellite data product requirements. This paper describes the features of HDF5, the current developments in HDF and how it can be useful as one of the output formats for IRS data products.

Overview

1. Why HDF?
2. Features of HDF File Format
3. HDF5
 - 3.1 HDF5 File Organisation and Data Model
 - 3.2 HDF5 Groups
 - 3.3 HDF5 Datasets
 - 3.4 HDF5 Attributes
 - 3.5 Physical File Structure
4. HDF-EOS
5. HDF5 as IRS Data Products Format

6. HDF5 Utilities
7. Conclusions
8. References

1. Why HDF?

In 1987 a group of users and developers at NCSA searched for a file format that would satisfy NCSA's data need. The design requirement of HDF was to create a scientific data format, which would be easy, straightforward, and self-describing means of sharing scientific data among people, projects, and different types of computers.

The basic requirement of scientific data management is the ability to store and access as much information in as many ways, as quickly and as easily as possible. A data storage and retrieval system that have these capabilities must provide following features:

- *Support for Scientific Data and Metadata* - storage of scientific data requires support for various datatypes, data sets (including images) that can be extremely large and complex. Metadata, supplementary data that describes the basic data (sometimes referred to as raw data), includes information such as the dimensions of an array, the datatype of the elements of a record or a color lookup table (LUT).
- *Support for a range of hardware platforms* – data can originate from any one machine only to be used later on many different machines. So the aim was to help scientists to access data and metadata on as many hardware platforms as possible.
- *Support for a range of software tools* – for searching, analyzing, archiving and transporting the data and metadata scientists need a variety of tools, utilities and a range of library for reading and writing.
- *Rapid Data Transfer* – adoption of proper compression technique to reduce data transfer time.
- *Extensibility* – new types of data generated by different scientific work must be accommodated in this format.

All the above requirements were considered before designing the HDF. The first version of HDF was implemented in 1988. The current HDF version is HDF5. The HDF5 related information on the Internet could be found in the URL [4].

2. Features of HDF File Format [NCSA, Jan. 2001]

- It is *versatile*. HDF supports several different data models. Each data model defines a specific aggregate datatype and provides an Application Programming Interface (API) for reading, writing, and organising data and metadata of the corresponding type. Data models supported include multidimensional arrays, raster images, and tables.
- It is *self-describing*, allowing an application to interpret the structure and contents of a file without any outside information.
- It is *flexible*. With HDF, one can mix and match related objects together in one file and then access them as a group or as individual objects. Users can also create their own grouping structures using an HDF feature called **vgroups**.
- It is *extensible*. It can easily accommodate new data models, regardless of whether they are added by the HDF development team or by HDF users.
- It is *portable*. HDF files can be shared across most common platforms, including many workstations and high performance computers. An HDF file created on one computer can be read on a different system without modification.

3. HDF5 [NCSA, July 2001]

The latest version of HDF, the HDF5, overcomes a number of limitations present in the earlier versions, which are listed below:

- A single file cannot store more than 20,000 complex objects, and a single file cannot be larger than 2 gigabytes.
- The data models are less consistent than they should be, there are more object types than necessary, and datatypes are too restricted.
- The library source is old and overly complex, does not support parallel I/O effectively, and is difficult to use in threaded applications.

Further, the following improvements have been incorporated in the new version:

- A new file format designed to address some of the deficiencies of HDF4.x, particularly the need to store larger files and more objects per file.
- A simpler, more comprehensive data model that includes only two basic structures: a multidimensional array of record structures, and a grouping structure.
- A simpler, better-engineered library and API, with improved support for parallel I/O, threads, and other requirements imposed by modern systems and applications.

3.1 HDF5 File Organisation and Data Model: HDF5 files are organized in a hierarchical structure, with two primary structures: *groups* and *datasets*.

- *HDF5 group*: a grouping structure containing instances of zero or more groups or datasets,

together with supporting metadata.

- *HDF5 dataset*: a multidimensional array of data elements, together with supporting metadata.

Working with groups and group members is similar in many ways to working with directories and files in UNIX. As with UNIX directories and files, objects in an HDF5 file are often described by giving their full (or absolute) path names.

Any HDF5 group or dataset may have an associated *attribute list*. An *HDF5 attribute* is a user-defined HDF5 structure that provides extra information about an HDF5 object.

3.2 HDF5 Groups: An *HDF5 group* is a structure containing zero or more HDF5 objects. A group has two parts:

- A *group header*, which contains a group name and a list of group attributes.
- A *group symbol table*, which is a list of the HDF5 objects that belong to the group.

3.3 HDF5 Datasets: A **dataset** is stored in a file in two parts: **a header** and **a data array**.

The header contains information that is needed to interpret the array portion of the dataset, as well as metadata (or pointers to metadata) that describes or annotates the dataset. Header information includes the name of the object, its dimensionality, its number-type, information about how the data itself is stored on disk, and other information used by the library to speed up access to the dataset or maintain the file's integrity.

The four essential classes of information in any header are **name**, **datatype**, **dataspace**, and **storage layout**.

- Name:** A dataset name is a sequence of alphanumeric ASCII characters.
- Datatype:** HDF5 allows one to define many different kinds of datatypes. There are two categories of datatypes: atomic datatypes and compound datatypes. Atomic datatypes can also be system-specific, or NATIVE, and all datatypes can be named:

- Atomic datatypes are those that are not decomposed at the datatype interface level, such as integers and floats.
- NATIVE datatypes are system-specific instances of atomic datatypes.
- Compound datatypes are made up of atomic datatypes.
- Named datatypes are either atomic or compound datatypes that have been specifically designated to be shared across datasets.

Atomic datatypes include integers and floating-point numbers. Each atomic type belongs to a particular class and has several properties: size, order, precision, and offset. Atomic classes include integer, float, date and time, string, bit field, and opaque. (Note: Only integer, float and string classes are available in the current implementation.)

NATIVE datatypes: Although it is possible to describe nearly any kind of atomic datatype, most applications will use predefined datatypes that are supported by their compiler. In HDF5 these are called *native* datatypes. NATIVE datatypes are C-like datatypes that are generally supported by the hardware of the machine on which the library was compiled. In order to be portable, applications should almost always use the NATIVE designation to describe data values in memory. The NATIVE architecture has base names, which do not follow the same rules as the others. Instead, native type names are similar to the C type names. Table 1 gives relation between HDF variables and C language variables.

Table 1: Examples of Native Datatypes and Corresponding C Types

Example	Corresponding C Type
H5T_NATIVE_CHAR	Signed char
H5T_NATIVE_UCHAR	Unsigned char
H5T_NATIVE_SHORT	Short
H5T_NATIVE_USHORT	Unsigned short
H5T_NATIVE_INT	Int
H5T_NATIVE_UINT	Unsigned
H5T_NATIVE_LONG	Long
H5T_NATIVE_ULONG	unsigned long
H5T_NATIVE_LLONG	long long
H5T_NATIVE_ULLONG	unsigned long long
H5T_NATIVE_FLOAT	float
H5T_NATIVE_DOUBLE	double
H5T_NATIVE_LDOUBLE	long double
H5T_NATIVE_HSIZE	hsize_t
H5T_NATIVE_HSSIZE	hssize_t
H5T_NATIVE_HERR	herr_t
H5T_NATIVE_HBOOL	hbool_t

A *compound datatype* is one in which a collection of several datatypes are represented as a single unit, similar to a *struct* in C. The parts of a compound datatype are called *members*. The members of a compound datatype may be of any datatype, including another compound datatype. It is possible to read members from a compound type without reading the whole type.

Named datatypes. Normally each dataset has its own datatype, but sometimes we may want to share a datatype among several datasets. This can be done using a *named* datatype. A named datatype is stored in the file independently of any dataset, and referenced by all datasets that have that datatype. Named datatypes may have an associated attributes list.

c) Dataspace: A dataset dataspace describes the dimensionality of the dataset. The dimensions of a dataset can be fixed (*unchanging*), or they may be unlimited, which means that they are extendable (*i.e. they can grow larger*).

Properties of a dataspace consist of the *rank* (number of dimensions) of the data array, the *actual sizes of the dimensions* of the array, and the *maximum sizes of the*

dimensions of the array. For a fixed-dimension dataset, the actual size is the same as the maximum size of a dimension. When a dimension is unlimited, the maximum size is set to the value H5P_UNLIMITED. (An example below shows how to create extendable datasets.)

A dataspace can also describe portions of a dataset, making it possible to do partial I/O operations on *selections*. *Selection* is supported by the dataspace interface (H5S).

d) Storage layout: The HDF5 format makes it possible to store data in a variety of ways. The default storage layout format is contiguous, meaning that data is stored in the same linear way that it is organised in memory. Two other storage layout formats are currently defined for HDF5: compact, and chunked.

3.4 HDF5 Attributes: Attributes are small named datasets that are attached to primary datasets, groups, or named datatypes. Attributes can be used to describe the nature and/or the intended usage of a dataset or group. An attribute has two parts: (1) a *name* and (2) a *value*. The value part contains one or more data entries of the same datatype.

The attribute API (H5A) is used to read or write attribute information. When accessing attributes, they can be identified by name or by an *index value*. The use of an index value makes it possible to iterate through all the attributes associated with a given object.

3.5 Physical File Structure: Though the physical file structure of HDF5 is irrelevant to general users, but it is helpful for specialized users. The HDF5 API generally exposes only the high-level elements to the user; the low-level elements are often hidden.

An HDF5 file appears to the user as a directed graph. The nodes of this graph are the higher-level HDF5 objects that are exposed by the HDF5 APIs, viz., groups, datasets, datatypes and dataspace. At the lowest level, as information is actually written to the disk, an HDF5 file is made up of the following objects: A super block, B-tree nodes (containing either symbol nodes or raw data chunks) and Object headers, Collections, Local heaps & Free Space.

Figure 1 depicts the relationship between the HDF5 root group, other groups and objects, while figure 2 depicts the structure of HDF5 objects.

The HDF5 library uses these lower-level objects to represent the higher-level objects that are then presented to the user or to applications through the APIs. For instance, a group is an object header that contains a message that points to a local heap and to a B-tree which points to symbol nodes. A dataset is an object header that contains messages that describe datatype, space, layout, filters, external files, fill value, etc., with the layout message pointing to either a raw data chunk or to a B-tree that points to raw data chunks.

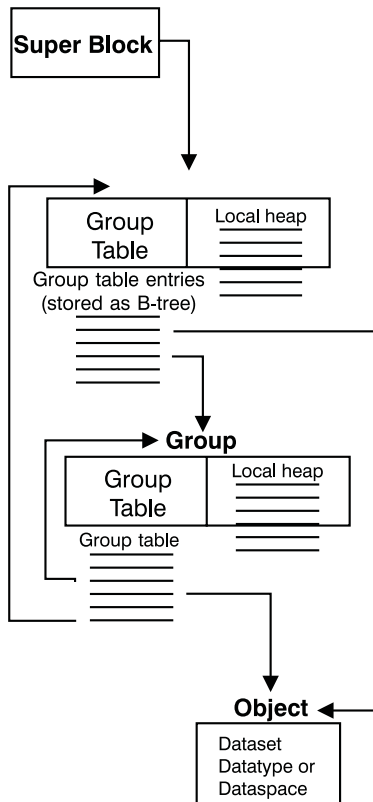


Figure 1: Relationship among HDF5 root group, other groups and objects

4. HDF-EOS

The HDF has been selected by the EOSDIS project as the format of choice for distribution of standard products. To bridge the gap between the needs of EOS data products and the capabilities of HDF, three new EOS specific datatypes – **point**, **swath**, and **grid** – have been defined within the HDF framework. Each of these new datatypes is constructed using conventions for combining standard HDF datatypes and is supported by a special API, which aids the data product user or producer, in the application of the conventions. The APIs allow data products to be created and manipulated in ways appropriate to each datatype, without regard to the actual HDF objects and conventions underlying them.

The sum of these new APIs comprises the HDF-EOS library. The *point* interface is designed to support data that has associated geo-location information, but is not organised in any well-defined spatial or temporal way. The *Swath* interface is tailored to support time-ordered data such as satellite swaths (which consist of a time-ordered series of scan lines), or profilers (which consist of a time-ordered series of profiles). The *Grid* interface is designed to support data that has been stored in a rectilinear array based on a well-defined and explicitly supported projection.

5. HDF5 as IRS Data Products Format

Since HDF5 is a general-purpose scientific data format, it has been chosen as one of the digital data product

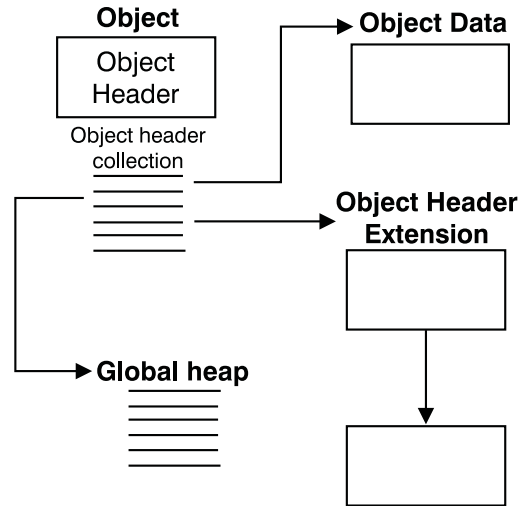


Figure 2: HDF5 Objects - datasets, datatypes or dataspace

formats for future IRS missions. IRS data products are supplemented with a lot of ancillary, ephemeris, attitude, calibration and mission related data along with digital image. These supporting data along with the digital image data can be well accommodated in HDF5. The new data types (Point, Grid, Swath) added by HDF-EOS are specifically very useful to store satellite data.

6. HDF5 Utilities

There are various utilities available to facilitate smooth transition of HDF4 to HDF5 and analysing HDF5 files. This section describes a few such utilities.

h5dump and **h5toh4**: The *h5dump* utility offers a convenient way to dump the contents of an HDF5 file. The *h5toh4* utility is a special-purpose tool developed for users who must use tools built on the HDF Release 4.x library to analyse files created with the HDF5 library.

h5utils: *h5utils* is a set of utilities for visualization and conversion of scientific data in the free, portable HDF5 format. Programs included are:

- *h5totxt* and *h5fromtxt*: convert HDF5 datasets to/from ASCII text (e.g. comma- or tab-delimited).
- *h5topng*: convert 2d slices of HDF5 datasets to PNG images, with a variety of color tables and other options.
- *h5tov5d*: convert HDF5 datasets to the format used by the free 3d+ visualization tool Vis5d.
- *h5tovtk*: convert HDF5 datasets to VTK format for use by the free Visualization Toolkit (along with supporting programs like MayaVi).
- *h5read.oct*: a plug-in for GNU Octave (a Matlab-like program) to read 2d slice of HDF5 datasets. (The latest versions of Octave also include native support for HDF5.)
- *h5fromh4*: convert HDF (version 4) datasets to HDF5; mostly superseded by the *h4toh5* program included with recent versions of HDF5.

7. Conclusion

Since HDF5 is a general-purpose scientific data format with potential to store and disseminate virtually all types of data, it is gaining wide spread popularity among scientific community. HDF-EOS with special data types like *Point*, *Swath* and *Grid* has added advantage in storing satellite data. HDF5 and HDF-EOS have been adopted by IRS-P6 (RESOURCESAT) as digital data product formats.

8. References

- [1] NCSA, HDF Specification and Developer's Guide, Version 4.1r4, January 2001.
- [2] NCSA, HDF5 User Documentation, Release 1.4.2, July 2001.
- [3] NCSA, HDF-EOS Interface Based on HDF5, Vol.1,2: March 2001.
- [4] <http://hdf.ncsa.uiuc.edu/HDF5/>