

Transition from HDF5 for HDF4 Users: status and goals

Robert E. McGrath
National Center for Supercomputing Applications
University of Illinois, Urbana-Champaign

December 5, 2002

Contents

1. Introduction.....	1
2. Four Important Goals.....	2
2.1. Supporting both HDF4 and HDF5 format and software.....	2
2.2. Interoperation of Data and Libraries.....	2
2.3. Conversion of Data.....	3
2.4. Conversion of Software.....	4
3. Suggestions For Users.....	5
3.1. If you haven't started, use HDF5 from the beginning.....	5
3.2. If you are using HDF4, should you start using HDF5?.....	5
3.3. Should I convert data or use two readers?.....	5
4. Goals for NCSA and NASA.....	6
4.1. Get HDF4 based software into "safe mode".....	6
4.2. Help for software developers.....	6
For More Information.....	7
Acknowledgements.....	7
References.....	7
Appendix 1: Why HDF5?.....	8
Appendix 2. Three Principles for understanding HDF5.....	8
<i>Do what makes sense</i>	8
<i>Think of HDF5 as a completely new file format</i>	8
<i>Anything you can do in HDF4, you can do in HDF5</i>	9

1. Introduction

HDF5 is a new format and library, designed based on lessons learned from HDF4, and implemented for contemporary systems. HDF5 is not backward compatible with HDF4. HDF5 is conceptually a superset of HDF4, with many new features. HDF5 has been in production use since 1998, and will be used in future NASA Earth Science data systems.

One of the important goals of the NCSA HDF group is to support and help users to transition from HDF4 to HDF5 (or to use HDF4 and HDF5 together). This note reviews the status of this effort and points to areas where further effort will be needed.

Most environments will be using both HDF- and HDF5-based software for many years. It is reasonable to expect that usage of HDF4 will slowly decline, and the use of HDF5 will rapidly grow over the next few years. At this time it is not a question of whether to use HDF4 or HDF5, but how to use both in the same environment.

While many users will continue to rely on HDF4, most new projects are selecting to use HDF5. Since there is an enormous amount of data and software in HDF4, it is clear that many

Transition to HDF5

environments will be using both formats for a very long time. For example, the current holdings of the NASA ESE use HDF4, but Aura data will be HDF5. Inevitably, many, if not most, NASA users and data systems will be using both HDF4 and HDF5.

Section 2 reviews goals and progress with respect to four important tasks. Section 3 presents suggestions for why, when, and how to port to HDF5. Section 4 discusses important areas for future work. Two appendices provide background information about the differences between HDF4 and HDF5.

2. Four Important Goals

This section briefly reviews four important tasks needed to support the transition from HDF4 to HDF5 (or, more likely, HDF4 and HDF5 together). First, it is and will remain critical to support both the HDF4 and HDF5 formats and libraries, and software based on these formats, especially HDF-EOS4 and HDF-EOS5. Second, it is important that the software and formats can be used together in systems that use both formats. Third, it is important to be able to convert data between HDF4 and HDF5 when needed, as well as is practical. And fourth, software must be adopted or ported to use HDF5 instead of or as well as HDF4.

2.1. Supporting both HDF4 and HDF5 format and software

Both HDF4 and HDF5 are supported by the NCSA HDF group. We will continue to maintain HDF4, as long as we are funded to do so. We do not plan to add any new features to HDF4, but we will fix bugs and build and test it on new operating system versions. We recommend using HDF5, especially if the project is not constrained to using HDF 4.x (HDF4). We also recommend that you consider migrating from HDF4 to HDF5 to take advantage of the improved features and performance of HDF5.

Maintaining two formats and libraries is fairly routine, although obviously requires more effort than supporting only one. In the case of HDF5, the software is actively maintained, and the challenges are to manage the evolution of the software.

In the case of HDF4, the code is aging and we will face increasing difficulty porting to new platforms. We should also expect the pool of expertise to erode as programmers age and depart. One of the critical goals for HDF4 must be to move it into a “safe-mode”, so it may be viable for a decade or more without intensive labor.

The NASA ESE is currently supporting HDF-EOS4 and HDF-EOS5 as well. This library will face similar challenges discussed for the HDF4 and HDF5 libraries, although the rate of change is likely to be very low.

It is important to note that in the case of HDF-EOS, the metadata is largely unaffected by the format change. For many tools, metadata handling will not need to change, and the EOS metadata should enable many tools to transparently use HDF-EOS4 and HDF-EOS5. To the extent that my claim is born out, this will be a strong validation of the value of the remarkable effort that has gone into the modeling, definition, and implementation of the EOS metadata.

2.2. Interoperation of Data and Libraries

If we accept that many if not most users will use both HDF4 and HDF5 (and HDF-EOS4 and HDF-EOS5) data in the same environment and programs, it is essential the formats and libraries

Transition to HDF5

can be used together, i.e., in the same program. This goal has been achieved, largely by making the formats and libraries completely separate, with separate name spaces for the library functions.

The key technique here is to separate the concepts of the “application data model”, i.e., the science and application tools, from the “storage data model”. The application and science concepts and data structures are unaffected by the data formats used. The application concepts should be mapped to a model of data storage which can then be implemented using alternative libraries and formats.

While this seems obvious, there have been a number of missteps on the road to this goal, and many application software layers must still work through these issues. In retrospect, it seems that one of the great advantages of a totally new format and library is that it doesn’t collide with the old, reducing versioning and compatibility problems in software that uses both libraries.

While HDF4 and HDF5 are highly interoperable today, it is and will become increasingly difficult to keep both libraries working on the same set of platforms, with the same set of compilers. For example, HDF4 has a Fortran77 interface and HDF5 has a Fortran90 interface. It will take very significant work to produce an F90 interface for HDF4. So, while HDF4 and HDF5 are supported for the same platforms, and both have a Fortran interface, it could be quite challenging for a Fortran application to actually use both libraries. This is only one example of the inevitable “genetic drift” we can expect as computing platforms evolve.

2.3. Conversion of Data

One approach to dealing with the heterogeneity of data is to write software to one format, and convert data to that format. Thus, one could try to convert data from HDF5 to HDF4, or from HDF4 to HDF5, or from one version of HDF-EOS to the other. (In general, the recommendation is to convert data from HDF4 to HDF5, and use HDF5 in the future.) Data conversion could be done on-demand (as needed), for whole collections, or during routine reprocessing and refresh.

To make this as easy as possible, NCSA has developed a toolkit of documentation, utilities, and library software to support conversion between HDF4 and HDF5 [2]. This work includes:

- A complete specification of the conceptual mapping of HDF4 objects to HDF5 objects [1].
- Utilities to implement this mapping to convert an HDF4 file to HDF5 or HDF4 to HDF5 (if possible) [3]
- A library to convert individual HDF4 objects and sets of objects from HDF4 to HDF5 [4]

The *Mapping HDF4 Objects to HDF5 Objects* specification provides conceptual guidance to users and developers [1]. It also provides recommended standards, which will increase interoperability of converted files. This specification is a general purpose solution, it should be customized for specific uses.

The *h5toh4* and *h4toh5* utilities implement the Mapping specification from [1], to copy all the objects of a file into another file [3]. Only some of the objects of HDF5 can be mapped to HDF4, so the *h5toh4* tool cannot convert all objects of all files. The *h4toh5* tool converts all the objects from an HDF4 file to default objects in an HDF5 file. For simple cases, this default conversion may be sufficient, but the resulting HDF5 file may not be an ideal use of HDF5.

The H4 to H5 library [4] provides an API and C library that implements the mapping specification for individual HDF4 objects and sets of objects. The H4 to H5 library gives more control over the conversion, and can be used to construct customized data conversion tools.

Transition to HDF5

In a similar vein, the *heconvert* utility converts the HDF-EOS objects in an HDF-EOS4 file to an equivalent HDF-EOS5 file [10]. In earlier work, it was shown that the H4 to H5 library can be used to convert the non-EOS objects in the HDF-EOS files [9].

The NCSA toolkit and similar software can provide “generic” conversion of the objects of HDF files. However, most HDF files, notably NASA ESE data products, have a complex and meaningful structure. The default conversion may reproduce the structure, but this may not be a good HDF5 file. In general, when data is stored in HDF5, the software will use new features and structures made possibly by the new format. In such a case, the conversion needs to be done by higher level software that is cognizant of design of the data.

HDF-EOS files are one example of this case. The *h4toh5* tool can read and convert all the objects of an HDF-EOS4 file into an HDF5 file. The result is a valid HDF5 file, but is not a valid HDF-EOS5 because HDF-EOS5 stores its objects in ways designed to use HDF5 optimally. To convert an HDF-EOS4 file, it is necessary to use the *heconvert* utility, which converts the EOS objects (Grids, etc.), not the HDF objects. (See [9] for a more detailed discussion.)

The default conversion may be undesirable for other reasons. For example, it is difficult or impossible to detect data which has not been written because the *read* returns fill values. As a result, the converted object may be filled with written-out fill values. Another difficult case is an HDF4 file that uses external files. This file appears to be a single file, and the converted HDF5 file will be one large file. This might be desirable, or might be disastrous, depending on the purpose of the original storage model. In the NASA ESE domain, Landsat 7 data is stored in HDF4 using external files.

Besides the layout of the files, it will be necessary to validate the conversion, to assure that the converted file faithfully reproduces the scientific content of the original file. I.e., even though the original file has been carefully validated, it may be necessary to re-validate the converted file to assure the quality of the data. This should be a relatively simple process for arrays of numbers and strings, but any complex linking structure (such as an HDF-EOS Swath) would need to be checked for integrity and validity.

The work to date has shown that data conversion from HDF4 to HDF5 is feasible, even for large collections [8,9]. The NCSA toolkit makes it fairly simple for any given case, e.g., a particular data product. But there are a very large number of cases, and each may need a custom conversion. Since the conversion is usually quite efficient, the development and validation of conversion software will be the biggest cost of data conversion. However, this is something that might naturally be done as part of reprocessing or periodic refresh.

2.4. Conversion of Software

An increasingly important challenge for the transition to HDF5 is the transition of libraries, applications, and user software. Again, we must assume that many if not most computing environments will use both HDF4 and HDF5 for many years to come. In the case of NASA ESE, it is nearly certain that HDF-EOS4 and HDF-EOS5 will be in use for a decade or two.

Therefore, one common and critical task will be adding HDF5 and HDF-EOS5 support to existing applications and environments. When HDF5 is considered a new format, this task becomes clear: the software needs to add new readers and writers for the new format. This is no more or less difficult than adding any other new format. In fact, this is already occurring in NCSA and commercial tools.

Transition to HDF5

While each case is unique, this process usually requires a conceptual mapping of the problem data to the new (e.g., HDF5) storage format. If the HDF5 format is already designed, this process determines how the objects in the HDF5 file are to be used in the application, and how application objects are to be stored. For example, data from one or more HDF5 Datasets or HDF-EOS Swaths will be read into arrays in memory, to be used in science algorithms.

In the case of HDF-EOS, the process is relatively easy because of the well defined data and metadata model. Given software that uses HDF-EOS4, it is relatively easy to add an HDF-EOS5 driver, because the data model is well defined and the HDF-EOS library manages the details of the libraries and storage. Also, as already pointed out, the HDF-EOS metadata is independent of the format, and further helps the application software to use the data correctly.

If the use of the HDF5 format has not been designed yet, then a storage model must be designed and then implemented. In this case, it is necessary to determine what application data should be stored and how it should be laid out. If HDF4 is already used, the design of the HDF4 storage can be used as the starting point. But it is important to examine the requirements of the application and to design the HDF5 storage to best advantage.

In general, this process should not be especially difficult in any given case, unless the data is very complex. However, every case is slightly different, so the task will be faced repeatedly. Also, there is no one right way to do this, so we may well see variations sufficiently different to be incompatible.

3. Suggestions For Users

This presents some recommendations for users and developers.

3.1. If you haven't started, use HDF5 from the beginning.

For a new project or software package, use HDF5.

The best reason to use HDF4 is to gain compatibility with existing data and software. If this reason doesn't apply or isn't critical, then just use HDF5.

3.2. If you are using HDF4, should you start using HDF5?

Clearly, it makes sense to use software and data that is already available and working, and to continue to use software and data that meet your needs.

But, as this paper argues, most environments will soon be using both HDF4 and HDF5. Granted this claim, the question is not "whether", but "when" and "how" to use HDF5.

3.3. Should I convert data or use two readers?

This paper has discussed two approaches to using HDF4 and HDF5 together. One way is to convert data to one format, preferably from HDF4 to HDF5. The rest of the system need only deal with one standard format. The second approach is to add reading and writing capabilities for HDF5 to existing software. (Note that many software packages already read and write many formats, so this is a straightforward extension of the general practice.) Each approach has advantages and disadvantages.

Transition to HDF5

Converting data requires development and maintenance of conversion software, and possibly significant overhead to perform the conversion on large amounts of data. Also, there may be multiple versions of the “same” file, the original and the converted one. This increases storage, bandwidth, and management costs. On the other hand, data consumer software will be simpler, requiring only one reader/writer.

Multiple readers/writers requires development and testing of the I/O modules, and any changes to the software may have to be implemented twice. In some cases, only a reader is needed, which is easier. Of course, it is only possible to add readers if the source code is available, the package can be extended, and someone has the resources to do the work.

The overall amount of effort to create a reader/writer or a converter is probably similar. Unfortunately, it may be necessary to create custom converters or readers for specific data products.

4. Goals for NCSA and NASA

This section presents some suggestions for important goals for the future.

4.1. Get HDF4 based software into “safe mode”

The HDF4 format and library will remain important for many years to come, but it will become increasingly difficult to maintain. To be useable in the future, it will be necessary to “port” to future systems, which might be considered a “temporal port”: the same environment, but jumping several or many years of system evolution. With continuous maintenance and periodic releases, this is an incremental process. Farther in the future, it may be necessary to bring the software up on a system that has never been supported, and to vault long gaps in software and hardware.

It would be very valuable to move the HDF4 library to a sort of “safe mode”, analogous to satellite systems. The goal would be to bring the software to a well-known “dormant” state from which it can be safely and correctly awakened.

Steps toward this goal might include:

- Document the format and library
- Clean up and document source code
- Porting guide for “temporal ports”

In the past two years, NCSA has exerted considerable effort to document the HDF4 format and library (especially, [11]). Much work could be done to stabilize and clean up the library, and document the information future maintainers will need.

4.2. Help for software developers

This paper has pointed out the increasing importance of updating existing software to support both HDF4 and HDF5 formats. There is no silver bullet for this problem, software evolution is difficult and slow. Steps that may be useful include;

- Documentation, training, and consulting for users and developers
- Toolkits to assist software conversion

The NCSA FAQ and these workshops are implementing the first. Toolkits for software conversion are very difficult, but we can think about what might be done.

Transition to HDF5

For More Information

Information and HDF software is available from:

<http://hdf.ncsa.uiuc.edu>

Questions and problems should be sent to:

hdfhelp@ncsa.uiuc.edu

Acknowledgements

This report is based upon work supported in part by a Cooperative Agreement with NASA under NASA grant NAG 5-2040 and NAG NCCS-599. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

Other support provided by NCSA and other sponsors and agencies (See: <http://hdf.ncsa.uiuc.edu/acknowledge.html>).

References

1. Mike Folk, Robert E. McGrath, and Kent Yang, *Mapping HDF4 Objects to HDF5 Objects*, National Center for Supercomputing Applications, University of Illinois, July, 2002. <http://hf.ncsa.uiuc.edu/doc/ADGuide/H5toH5Mapping.pdf>
2. "HDF (4.x) and HDF5", <http://hdf.ncsa.uiuc.edu/h4toh5/>
3. "HDF5 Tools", <http://hdf.ncsa.uiuc.edu/HDF5/doc/Tools.html#Tools-Dump>
4. "H4toH5 Conversion Library", <http://hdf.ncsa.uiuc.edu/h4toh5/libh4toh5.html>
5. "HDF5: High Level APIs", http://hdf.ncsa.uiuc.edu/HDF5/hdf5_hl/doc/
6. "Image and Palette Specification", <http://hdf.ncsa.uiuc.edu/HDF5/doc/ADGuide/ImageSpec.html>
7. "HDF5 Table Specification", http://hdf.ncsa.uiuc.edu/HDF5/hdf5_hl/doc/RM_hdf5tb_spec.html
8. Robert E. McGrath, "Tests of the NCSA h4toh5 utility with NASA Datasets, August 2001. <http://hdf.ncsa.uiuc.edu/h4toh5/Experiment/>
9. Robert E. McGrath and Muqun Yang, "Conversion of from HDF4 to HDF5: 'Hybrid' HDF-EOS Files", <http://hdf.ncsa.uiuc.edu/h4toh5/Experiment2/>
10. HDF-EOS Tools and Information Center, *heconvert*, <http://hdfeos.gsfc.nasa.gov/hdfeos/details.cfm?swID=6>
11. "HDF Specification and Developer's Guide v4.1r5", <ftp://ftp.ncsa.uiuc.edu/HDF/HDF/Documentation/HDF4.1r5/Spec/>

Transition to HDF5

Appendix 1: Why HDF5?

Versions 1-4 of HDF maintained backward compatibility with earlier versions of the format and library, dating back to 1988. Why is HDF5 a new and incompatible format and library?

Despite the success and widespread use of HDF4, analysis of HDF4 showed a number of shortcomings, including:

- limits on object and file size (< 2GB)
- limits on the number of objects in a file (< 20K)
- rigid data models
- I/O performance issues

In addition, new demands and requirements have emerged, including:

- Bigger, faster machines and storage systems
 - Massive parallelism, teraflop speeds
 - Parallel file systems, terabyte storage
- Greater complexity and power features
 - Complex data structures (including variable length data, meshes, etc.)
 - Complex subsetting, sampling, etc.
 - Support for data transformations
 - Easier to extend, e.g., with custom compression
- Emphasis on remote and distributed access

After careful study, it was clear that these shortcomings and requirements could not be met with the HDF4 format and library. It was necessary to start fresh, to build a new format and library based on the lessons learned from HDF4.

Appendix 2. Three Principles for understanding HDF5

Do what makes sense

The most important guiding principle is “*do what is best for your needs*”. The HDF documentation, software, and tutorials give standard defaults, but these are not hard and fast rules. In many cases, there will be more than one possible approach, and in some cases it will be best to use HDF5 in a totally different way than HDF4 was used. *Do what makes sense for you.*

Think of HDF5 as a completely new file format

While HDF Versions 1-4 were backward compatible (both format and APIs), HDF5 is a new format and library. In fact, it is best to think of HDF5 as a completely different format and library. HDF5 data cannot be read or written by HDF4. HDF4 data cannot be read or written by HDF5. In general, it will be necessary to rewrite software that uses HDF4.

Thus, the task of adopting HDF5 is similar to porting to another format; it is necessary to learn the new software and adapt old software to use the new.

Transition to HDF5

Anything you can do in HDF4, you can do in HDF5

While HDF5 is new, it is highly compatible with HDF4. In fact, HDF5 is a conceptual superset of HDF4: in general, it is reasonable to expect that anything you can do with HDF4 you can do with HDF5; plus a lot more.* See [2].

* There are a few things that HDF4 can do that are not yet available for HDF5, including reading netCDF, and compression other than GZIP.