Mapping HDF4 Objects to HDF5 Objects

Mike Folk and the HDF Group
National Center for Supercomputing Applications, University of Illinois
February, 2000

Note to reader: We present here some guidelines on how to represent HDF4 objects in HDF5 and how to interpret HDF5 objects as HDF4 objects. It is meant to help in implementing software that has to deal with both formats in some consistent way, such as converting HDF4 files to HDF5, or adapting HDF4 tools to HDF5. It is very possible that we have missed some things, and gotten some things wrong, so please send any comments, questions and suggestions to the author at mfolk@ncsa.uiuc.edu*.*

1 Introduction

All versions of NCSA HDF from HDF1 through HDF4 are essentially the same. The HDF4 format and library are backwardly compatible with all earlier versions of HDF. HDF5 is different. Although it shares many features with earlier versions of HDF and is intended for essentially the same uses, HDF5 is a completely new file format, and the NCSA HDF5 API and library are also new and entirely different.

Many applications have been written for accessing, visualizing, and otherwise dealing with HDF4 objects and files. Few have as yet been written for HDF5. A great deal of development time and expense could be saved if some HDF4 applications could be adapted for dealing with HDF5 objects. The purpose of this paper is to facilitate such adaptations by establishing standard ways to (a) represent HDF4 objects in HDF5, and (b) interpret HDF5 objects *as* HDF4 objects.

Case (a) assumes that an application writes an HDF5 object intending for the object to be understood as a particular HDF4 object. It may add extra attributes to the object to make it conform as fully as possible to the corresponding HDF4 data model. Case (b) assumes that the HDF5 object was not created with HDF4 in mind, but nevertheless conforms to one or more HDF4 objects. (In the context of this paper, the term "conform" means that the characteristics of the HDF5 object are such that it would be meaningful and useful in to and HDF4 application. It does not mean that the HDF5 object is exactly the same as a corresponding HDF4 object would be.)

It is not our intention to map all possible HDF4 objects into HDF5, and vice versa. In section 2 we identify those HDF4 and HDF5 objects that will be mapped.

Section 2 describes the types of HDF4 and HDF5 objects that can be mapped, and identifies those for which mappings are not recommended.

Section 3 covers case (a), how to explicitly represent HDF4 objects in HDF5. It describes the basic mappings that are recommended, and and presents a rigorous set of rules to instantiate the mappings

Section 4 covers case (b), how to interpret HDF5 objects as if they were HDF4 objects.

Section 5 covers other considerations, such HDF4 reference numbers and file-level information.

In the following treatment, it is assumed that the reader is familiar with both HDF4 and HDF5 datatypes. The following documentation is available to provide this background information:

- HDF4 documentation: http://hdf.ncsa.uiuc.edu/doc.html
- HDF5 documentation: http://hdf.ncsa.uiuc.edu/HDF5/
- HCR documentation: ftp://ftp.ncsa.uiuc.edu/HDF/pub/HCR/Doc/HCR-Definitions/

2 HDF4 objects and HDF5 objects

The HDF4 format and library support the following eight basic objects:

- Scientific dataset (SDS), a multidimensional array with dimension scales
- 8-bit raster image (RIS8), a 2-dimensional array of 8-bit pixels

- 24-bit raster image (RIS24), a 2-dimensional array of 24-bit pixels
- General raster image (GR), a 2-dimensional array of multi-component pixels
- 8-bit color lookup table (palette), a 256 by 3 array of 8-bit integers
- Table (Vdata), a sequence of records
- Annotation, a stream of text that can be attached to any object
- Group, a structure for grouping objects

The HDF4 format also includes "primitive" objects that are used to construct these basic objects within an HDF4 file. These primitive objects are identified by "tags" within an HDF4 file. Since most HDF4 primitive objects have no counterpart in HDF5, nor are they accessed directly by most HDF4 users or applications, they will not be considered here. Exceptions to this are the HDF4 palette and annotation, which will be considered.

HDF5 supports two primary objects:

- Dataset, a multidimensional array of records
- Group, a structure for grouping objects

HDF5 also supports "attributes", which are (usually) small, named datasets that are associated with groups or datasets. HDF5 supports other objects, such as named datatypes, but these have no counterparts in HDF4 and hence will not be considered here.

3 Representing HDF4 objects in HDF5

In this section we provide detailed rules for representing HDF4 objects in HDF5. All eight basic HDF4 objects can be represented in HDF5. Usually such representations require restrictions or extra metadata. In the table 1, a mapping is shown from HDF4 objects to their HDF5 counterparts.

Table 1. Representing HDF4 objects in HDF5..

HDF4 object	**Corres-ponding HDF5 object**	**Restrictions**
SDS	Dataset	HDF4 DIMENSIONLIST becomes an HDF5 attribute, if it exists. HDF4 dimension scales become HDF5 datasets. Only the first dimension can be unlimited. Not all HDF4 storage properties are supported.
Image	Dataset	The HDF5 dataset must be 2-dimensional. If the number of pixel components is 1, an HDF5 scalar datatype is used, otherwise a compound type is used. If a palette is present, HDF5 attributes are used to indicate this. Not all HDF4 storage properties are supported.
Palette	Dataset	The HDF5 dataset must be a 256 by 3 array of 8-bit integers. HDF5 attributes describe this dataset as a standard 8-bit palette.
Vdata (table)	Dataset	The HDF5 dataset must be 1-dimensonal, with a compound datatype equivalent to corresponding HDF4 field and record structure. Non-interleaved fields are not permitted in HDF5. (This last restriction could be lifted if a structure is created to store fields as separate datasets.)
Annotation	Attribute	HDF4 *file* annotations are attributes of the HDF5 root group. HDF4 *object* annotations are attributes of the corresponding HDF5 object. Only annotations on the HDF4 objects listed here are supported.
Vgroup	Group	

As indicated in table 1, in all cases except Vgroups and annotations HDF4 objects are mapped to HDF5 datasets with simple dataspaces. Vgroups are mapped to HDF5 groups, and annotations are mapped to HDF5 attributes. In the tables that follow we identify all components of an HDF4 object that an application is likely to use, and map it to a corresponding HDF5 component. This mapping includes only persistent objects and components. Items that are available only when accessing HDF4 files (e.g. file id and object index) are omitted.

All of the HDF5 objects except annotations have the following two optional attributes: HDF4_OBJECT_TYPE and HDF4_REF_NUM. HDF4_OBJECT_TYPE can be used to tell applications that the object is compatible with an HDF4 object. HDF4_REF_NUM is available for those applications that use reference numbers as identifiers for HDF4 objects.

In HDF4, SDS, images, Vdatas and Vgroups have unique names. In the mapping, we use a "HDF4_OBJECT_NAME" attribute for this.

When translating an HDF4 object to HDF5, it may be possible to determine certain storage properties used for storing the HDF4 object. For example, valid storage properties for an SDS are compression, chunking, and external storage. If special storage properties are used in HDF4, and if those storage properties are available in HDF5, then they should be used to store the corresponding HDF5 object. For instance, if an HDF4 SDS is chunked, then the corresponding HDF5 datasets should be chunked in the same way. If it is not possible to determine an HDF4 storage property, then of course the HDF5 can stored without applying that property.

3.1 The mapping tables

In the following sub-sections, each of the six mappings from table 1 is described in detail with a table containing five columns:

- Column 1: a flag indicating whether the object is required ("R) in HDF5 in order for the object to conform to the corresponding HDF4 object. "O" (optional) means that it is not required.
- Column 2: components from HDF4 that are to be mapped to HDF5. Items with HDF names are in bold caps. Items in parentheses refer to information that is needed in the HDF5 version but do not have an HDF4 counterpart.
- Column 3: the HDF5 object that is mapped to.
- Column 4: information about the datatype, value, etc. of the HDF5 object.
- Column 5: additional information on how to perform the mapping.

The HCR definition of HDF5 was used to identify the HDF4 items that are to be mapped. In the tables, we have tried to used the HCR terminology whenever possible. For instance, in column 2, DATATYPE refers to an HDF4 datatype. Non-terminals are shown in angle brackets (e.g. <name>). Most non-terminals are defined in the HCR documentation. Others that are used are:

- <string>: any legal quoted string
- <name>: any valid name
- <value>: any valid scalar value
- <HDF4 datatype>: any valid HDF4 datatype.
- <uint16>: a value of type DFNT_UINT16

3.2 SDS

The SDS mapping requires two types of HDF5 dataset, one for the SDS array and one for dimension scales. For each dimension, the creator of the HDF5 "SDS" has create a corresponding dimension dataset with a unique name. In the HDF5 "SDS" an attribute "DIMENSIONLIST" is created consisting of pointers to each of the respective dimension datasets.

	HDF4 object	**HDF5 object or component**	**Datatype, value, etc.**	**Notes**
R	<SDSArray>	Dataset		Objects with unlimited dimensions are stored using chunked storage.

O	<SDS Dimension with Name>	Dataset		
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE = "SDS"	
O	<File annotation>	Attr		<User-defined attribute > in root group.

	<SDSArray>			
R	**NAME**	Attr	HDF4_OBJECT_NAME = <SDSArrayName>	
R	**DATATYPE**	Datatype	<HDF4 datatype>	
R	**DIMENSIONRANK & DIMENSIONSIZE**	Dataspace		dimension sizes are also part of dimension information.
R	**DIMENSIONLIST**	Attr	HDF4_DIMENSIONLIST = {ptr1, pt2, … ptrn}	Dimension names are <DimName1, <DimName2>, …,<DimNameN> as defined in HCR. See note 4.
R	(Data)	Data		How should "native" number types in HDF4 be handled?
O	<User-defined attribute >	Attr		rank = 1; size is fixed. Global attributes: see note 3.
O	<SDS pre-defined attribute >			
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
O	(Storage properties)			
O	Compression property	Storage prop		Use if supported in HDF5.
O	Chunk property	Storage prop		Use if supported in HDF5.
O	External storage	Storage prop		Use if supported in HDF5.

O	<User-defined attribute >	Attr	<name> = <value>*	rank = 1; size is fixed. Global attributes: see note 3.
O	**NAME**	Name	<name>	
O	**DATATYPE**	Datatype	<HDF4 datatype>	
O	**N_VALUES**	Num-values		
O	**DATA**	Data	<value>*	

O	<SDS pre-defined attribute >			Same names, datatypes, etc., as the hdf4 counterpart
O	**LONGNAME**	Attr	.	
O	**UNIT**	Attr		
O	**FORMAT**	Attr		
O	**COORDINATE_SYSTEM**	Attr		
O	**RANGE**	Attr		
O	**FILL_VALUE**	Attr		Note 1.
O	**SCALE_FACTOR**	Attr		
O	**SCALE_FACTOR_ERROR**	Attr		
O	**ADD_OFFSET**	Attr		
O	**ADD_OFFSET_ERROR**	Attr		
O	**CALIBRATED_NT**	Attr		

	<SDS Dimension with Name>	Dataset		
R	**NAME**	Name	<name>	
R	**SIZE**	Dataspace		rank=1. Only the first dimension can be unlimited.
O	**DATATYPE**	Datatype	<HDF4 datatype>	
O	**DATA**	Data	<value>*	
O	<Dimension pre-defined attribute>			These are dims of dim scale dataset, not SDS dataset
O	**LONGNAME**	Attr		How about named vs. unnamed dimensions?
O	**UNIT**	Attr		
O	**FORMAT**	Attr		
O	<User-defined attribute>	Attr		defined above

Note 1. The Fill-Value, if not explicitly defined, has default values. There are plans to support the Fill-Value feature as an HDF5 storage property, but it is not yet been fully defined.

Note 2. Dimension scales are to be stored in HDF5 as separate datasets. Hence, all of the information in this category is stored as part of the corresponding HDF5 dimension scale dataset. Dimension scale datasets are identified by the attribute DIMENSIONLIST in the SDS dataset.

Note 3. Global SDS attributes should be stored as attributes to the HDF5 root group.

Note 4. Dimensions must have names. If a dimension in an SDS does not have a name, one must be created in order to store the SDS in HDF5.

3.3 Vdata

Vdatas are mapped to HDF5 datasets of 1 dimensional extendable of compound datatype.

	Vdata	**HDF5 object or component**	**Datatype, value, etc.**	**Notes**
R	**NAME**	Attr	<name> = <string>	<name> is the same as a normal Vdata name
O	**CLASS**	Attr	HDF4_VDATA_CLASS = <string>	
O	**INTERLACEMODE**	NA		Full interlace always used in HDF5 version.
R	(Number of records)	Dataspace		Rank=1, curr_size = the number of records.
R	(Record)	Compound datatype		
R	(Field)	Member		Compound datatype member
R	**NAME**	Name	<name>	
R	**DATATYPE**	Datatype	<HDF4 datatype>	
R	**ORDER**	Num-values		What's the correct name for the size of the array?
O	<User-defined attribute>	Attr	<FieldName>:<name> = <value>	<FieldName> is value of **NAME** for the field.
R	(Data)	Data		
O	<User-defined attribute >	Attr	<name> = <value>	rank = 1; size is fixed;
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE =	

			"Vdata"	
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
O	(External storage)	Storage prop		

3.4 Vgroup

Vgroups are mapped to HDF5 group.

	Vgroup	**HDF5 object or component**	**Datatype, value, etc.**	**Notes**
R	**NAME**	Attr	HDF4_OBJECT_NAME = <string>	<name> is the same as a normal Vgroup name
O	**CLASS**	Attr	HDF4_VGROUP_CLASS = <string>	
R	<Vgroup member>	Group member		HDF5 hard link
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE=" Vgroup"	
O	<User-defined attribute>	Attr	<name> = <value>	rank = 1; size is fixed;
O	(Reference number)	Attr	REF_NUM = <uint16>	

Note: HCR defines three additional items: MEMBERTYPE, MEMBERNAME, and PALETTEINDEX. It would be awkward to represent these in HDF5 groups, and hence they have been omitted. If it is found that they are needed, they will be added later.

3.5 Image

Raster images (8-bit, 24-bit, and general raster (GR)) are mapped to HDF5 datasets with simple 2D dataspaces. Each element of the dataset is a one-dimensional array of pixel components.

	Image	**HDF5 object or component**	**Datatype, value, etc.**	**Notes**
R	**NAME**	Attr	HDF4_OBJECT_NAME = <string>	<name> is the same as a GR name
R	(Pixel type)	Datatype		If N_COMPS=1, use atomic, else compound with 1 field(?)
R	**N_COMPS**	Num values		order of field, if N_COMPS > 1
R	**COMP_TYPE**	Atomic type	<HDF4 datatype>	
R	**DIMENSIONSIZE**	Dataspace		rank=2
R	(image array)	data		
O	<User-defined attribute>	Attr	<name> = <value>	rank = 1; size is fixed;
O	(Class)	Attr	CLASS = "IMAGE"	
O	(HDF4 object type)	Attr	(HDF4_OBJECT_TYPE="raster8", "raster24" or "GR") or (CLASS = "IMAGE")	
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
O	<Image palette>		IMAGE_PALETTE = {ptr1, pt2, … ptrn}	each pointer points to a palette
O	(Image subclass)	Attr	IMAGE_SUBCLASS = "IMAGE_INDEXED"	Indicates a palette is to be used
O	(Storage properties)			
O	Compression	Storage prop		If supported in HDF5. JPEG and

				RLE are not supported in HDF5.
O	Chunking	Storage prop		
O	External storage	Storage prop		

Note: The HDF5 image conventions support additional information that is not supported in HDF4, such as image transparency.

3.6 Palette

Palettes are mapped to HDF5 datasets that are 2-D arrays of bytes with dimensions 256 x 3.

	Palette	**HDF5 object or component**	**Datatype, value, etc.**	**Notes**
R	(Datatype)	Atomic datatype	<uint8>	
R	(Data)	Data		
R	(Rank & dimension sizes)	Dataspace		rank=2, dimension size = 256x3 (See note 2.)
R	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE=" palette" or (CLASS = "PALETTE" and PAL_TYPE = "STANDARD8")	
O	<User-defined attribute>	Attr	<name> = <value>	rank = 1; size is fixed;
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
O	(Class)	Attr	CLASS = "PALETTE"	
O	(Palette type)	Attr	PAL_TYPE = "STANDARD8"	

Note 1: The HDF5 palette conventions support additional information that is not supported in HDF4, such as color model and range index.

Note 2: The HDF5 palette specification requires that a palette dataset have dimensions (`nentries` by `ncomponents`), where '`nentries`' is the number of colors (in this case 256) and '`ncomponents`' is the number of values per color (in this case 3).

3.7 Annotation

Annotations are mapped to HDF5 attributes. There are four kinds of HDF4 annotations: file labels and descriptions, and object labels and descriptions. File annotations will be attributes of the root group. Although object annotations can be associated with any tag/ref supported by HDF4, this specification supports only object annotations that are associated with HDF4 SDS, Vgroups, Vdatas, images, and palettes. Annotations will have the following HDF5 attribute names: FILE_LABEL<n>, FILE_DESCRIPTION<n>, OBJECT_LABEL<n>, OBJECT_DESCRIPTION<n>, where <n> is an integer used to distinguish one annotation from another. For instance, if an object had two object labels, the corresponding attribute names would be OBJECT_LABEL1 and OBJECT_LABEL2.

4 Interpreting HDF5 objects when there is not explicit metadata

What if an HDF4-based application encounters and object in an HDF5 file that does not contain the metadata described in the previous section? In this case, three possible outcomes can occur:

1. No HDF4 counterpart exists
2. The ambiguous case: there are more than one possible HDF4 counterparts to the HDF5 object
3. The HDF5 object has an unambiguous corresponding HDF4 counterpart

4.1 Case 1: No HDF4 counterpart

In the previous section, we indicated that HDF5 datasets, groups, and attributes might have HDF4 counterparts. All other HDF5 objects should be assumed not to have an HDF4 counterpart. For instance, the HDF5 "named datatype" object has no HDF4 counterpart.

But certain HDF5 datasets, attributes and groups also have no HDF4 counterpart, including:

- Any HDF5 dataset or attribute whose datatype is not equivalent to an HDF4 datatype. HDF4 datatypes include unsigned and signed 8-, 16-, 32- and 64-bit integers, and 32- and 64-bit IEEE floats.
- Datasets with datatypes of multiplicity greater than 1, unless they map to Vdatas.
- Any HDF5 object whose size is greater than 2^31-1.
- Any HDF5 attribute that is associated with an HDF5 dataset whose HDF4 counterpart is a palette.
- Any HDF5 soft link that does not point to a corresponding HDF4-compatible object.

4.2 Case 2: The ambiguous case

There are a number of cases where an HDF4 object could map to any of a number of HDF5 objects. Here are some such cases.

4.2.1 HDF4 "datasets"

According to Table 1 HDF5 datasets can represent SDS, images, palettes, and Vdatas. This can lead to certain ambiguities. For example, a 2-D HDF5 dataset of some HDF4-compatible datatype could be converted either to an HDF4 SDS, GR raster, or Vdata. We propose the following convention to resolve this ambiguous case:

Unless there is metadata to indicate otherwise, an HDF5 datasets with an HDF4-compatible scalar datatype is assumed to be an HDF4 SDS.

4.2.2 The root group

Another ambiguous case is the HDF5 root group, which has no precise counterpart in HDF4. The HDF5 root group could always be mapped to a corresponding HDF4 Vgroup, with all HDF4 objects descending from that group, but this might in some cases create a view that was unnatural for a particular application. It might be more natural, for instance, to ignore the root group and to treat all of its attributes as HDF4 file annotations. Therefore, with respect to the root group, we propose:

An application can treat an HDF5 root group in whatever way best fits with its view of HDF4 files.

4.2.3 Strings

HDF4 does not have a "string" datatype, so it is common to use a 1-D array, or a Vdata field of order greater than 1 to hold data that is meant to represent a character string. As a general rule,

An HDF4 structure that is intended to be a string, should map to the HDF5 string datatype.

There will be cases when it is not possible to know whether an HDF4 data structure was meant to be a string. In such cases, it is left to the application to choose a corresponding HDF5 representation.

4.3 Case 3: The mapping is unambiguous

Any HDF5 dataset that is not covered in case 1 or case 2 should map to and HDF4 SDS or Vdata. HDF5 groups map to Vgroups.

Any HDF5 dataset or group attribute becomes an attribute to the corresponding HDF4 object, if the object supports attributes. Otherwise it is interpreted as an annotation for the corresponding HDF4 object.

5 Other considerations

Most applications that deal with HDF4 files deal with the eight basic objects, but there is other information that sometimes must be considered.

5.1.1 Dealing with HDF4 reference numbers

Reference numbers should not, in general, be used by applications as identifiers for HDF4 objects. Nevertheless, since some applications use reference numbers in this way, it would be useful to have an unambiguous way to store equivalent identifiers with HDF5 objects. Hence, if an application must use a reference number in connection with an HDF5 object we propose using an attribute "REF_NUM" of type uint16 to indicate that the corresponding object is to be interpreted as having the given reference number. It is the responsibility of the application to provide a method of assigning valid values for such reference numbers.

5.1.2 File-level information

HDF4 and HDF5 both have certain file-level information. This includes information that is stored in the file, such as version number, and information about the file, such as its size. It is recommended that the following types of file characteristics be treated as indicated:

- Version number: the library version number of the HDF5 file should be ignored.
- File size: HDF5 files that are larger than 2^31-1 cannot be fully represented in HDF4. It is the responsibility of the application to decide how to deal with HDF5 files that are larger than 2^31-1.
- HDF5 user-defined header: to be treated as an HDF4 file description.
- HDF5 offset datatype and other datatypes: should be ignored.