

Mapping HDF4 Objects to HDF5 Objects

Version 4

Mike Folk¹, Robert E. McGrath², Kent Yang¹

Revised: October, 2000; July, 2002, August, 2003, September 2017

¹The HDF Group

²National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign

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. Although the latest revision is more than 10 years after the previous one, we feel that most information in the previous revision is still valuable. The latest revision corrects errors and updates the out-of-dated information. The significant improvement is to represent SDS dimension scales with the HDF5 Dimension Scales. Please send comments and corrections to hdfhelp@hdfgroup.org.

1 Introduction

All versions of HDF from HDF1 through HDF4 are essentially the same. The HDF4 format and library are backward 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, the HDF5 API and library are entirely different.

Many applications have been written for accessing, visualizing, and otherwise dealing with HDF4 objects and files. 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 to an 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.

It is also not our intention to cover all possible cases for mapping HDF4 to HDF5. This is a specification of recommended default mappings only. If a different mapping is more appropriate for a particular application, then it should be used.

The most important guiding principle is “*do what is best for your needs*”. 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.*

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

Sections 3-5 cover case (a), how to convert HDF4 files to HDF5.

Section 3 recommends ways to explicitly represent HDF4 objects in HDF5. It describes the basic mappings that are recommended, and presents a set of rules to instantiate the mappings, including metadata attributes that make an object conform as fully as possible to the corresponding HDF4 data model.

Section 4 covers issues related to overall file organization, including how to map the organization and name structure of an HDF4 file into HDF5.

Section 5 covers other considerations, such as how to deal the HDF4 reference numbers and file-level information.

Section 6 and 7 deal with case (b), how to interpret HDF5 objects as if they were HDF4 objects when there is no explicit metadata. Section 7 covers some specific issues that arise when converting HDF5 objects into HDF4 objects.

This document relies on a number of documents, which the reader may need to refer to for background concerning HDF4 and HDF5 datatypes. This includes the HDF4 User's Guide, the HDF4 Specification and the HDF5 documentation cited in the reference section at the end of this document [2][3]. Also cited in the Reference section is the "HDF Configuration Record" specification [1], which provides a rigorous definition of all HDF4 objects.

Readers familiar with HDF4 but unfamiliar with HDF5 may wish to read the tutorial, *HDF5 for HDF4 Users: a short guide* [7].

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
- VGroup, 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 includes two primary objects:

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

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

Classes of objects can be defined as special cases of the basic HDF5 Dataset. To date, specifications have been created for raster images [4], tables similar to HDF4 Vdata objects [5] and dimension scales similar to HDF4 SDS dimension scales[6]. These specifications define conventions for storing a raster image, palette, one-dimensional table and dimension scales in HDF5 datasets. The specifications give conventions for using specific HDF5 attributes to describe the objects. These conventions should be used when converting HDF4 objects to HDF5.

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 Table 1, a mapping is shown from HDF4 objects to their HDF5 counterparts.

Table 1. Representing HDF4 objects in HDF5.

HDF4 object	Corresponding HDF5 object	Restrictions
SDS	Dataset	Only the first dimension can be unlimited. Not all HDF4 storage properties are supported. HDF4 dimension scales become HDF5 datasets. We recommend that the HDF5 object conform to the HDF5 Dimension Scale Specification [6]. Since the HDF5 Dimension Scale Specification had not been written at the time when version 3 of this document was out, this mapping document instructed to map HDF4 dimension scales differently then. The HDF4 to HDF5 conversion software was implemented following the version 3 of this document. To help users of the HDF4 to HDF5 conversion software understand the transition, we still keep the old dimension scale mapping information at Appendix A.
Image	Dataset	If the number of pixel components is 1, a 2-dimensional dataset is used. Otherwise a 3-dimensional dataset is used, with one dimension as the pixel components. If a palette is present, HDF5 attributes are used to indicate this. Not all HDF4 storage properties are supported. The HDF5 object should conform to the <i>HDF5 Image and Palette Specification</i> [4].
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. The HDF5 object should conform to the <i>HDF5 Image and Palette Specification</i> [4].
Vdata (table)	Dataset	The HDF5 dataset must be 1-dimensional, 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.) The HDF5 object should conform to the <i>HDF5 Table specification</i> [5].
Annotation	Attribute	HDF4 <i>file</i> annotations are attributes of the HDF5 root group. HDF4 <i>object</i> annotations are attributes of the corresponding HDF5 object. Only annotations on the HDF4 objects listed here are supported.
Vgroup	Group	HDF4 objects do not have to belong to any Vgroup, and there may be a forest of Vgroups in an HDF4 file. HDF5 is a rooted, directed graph of Groups, every object is a member of at least one group, except for the root group. Every HDF5 file has a root group.

As indicated in Table 1, in all cases except Vgroups and annotations HDF4 objects are mapped to HDF5 datasets with simple dataspace. 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.

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 HDF4 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 HDF4 was used to identify the HDF4 items that are to be mapped [1]. In the tables, we use 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`

3.1 SDS

An HDF4 SDS is mapped to an HDF5 Dataset. In version 3, the SDS Dimension Scales are mapped to HDF5 datasets following the mapping described in Appendix A. In this revision, the SDS Dimension Scales are mapped to HDF5 datasets by following the HDF5 Dimension Scale Specification [6].

Table 2. SDS mapping

	HDF4 object	HDF5 object or component	Datatype, value, etc.	Notes
R	<code><SDSArray></code>	Dataset		Objects with unlimited dimensions are stored using chunked storage.
O	<code><SDS Dimension with Name></code>	Dataset		(See note 1)
O	(HDF4 object type)	Attr	<code>HDF4_OBJECT_TYPE = "SDS"</code>	

	<SDSArray>			
R	NAME	Attr	HDF4_OBJECT_NAME = <SDSArrayName>	See Section 4 for details on how NAME is used as a link in HDF5.
R	DATATYPE	Datatype	<HDF4 datatype>	
R	DIMENSIONRANK & DIMENSIONSIZE	Dataspace		Dimension sizes are also part of dimension information.
R	DIMENSION_LIST	Attr	DIMENSION_LIST = {object_ref1, object_ref2, ... object_refn}	An array of object references that refer to the corresponding dimension datasets. This attribute should be generated by following the HDF5 Dimension Scale specification. The software implementation should use the HDF5 High-level Dimension scale APIs.
R	(Data)	Data		See section 0 for details on how to handle datatypes.
O	<User-defined attribute >	Attr		rank = 1; size is fixed. Global attributes: see note 2.
O	<SDS pre-defined attribute >	Attr		
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		rank = 1; size is fixed. Global attributes are covered below.
R	NAME	Attr name	<AttributeName>	
R	DATATYPE	Datatype	<AttributeType>	
R	N_VALUES	Num-values	<AttributeCount>	
R	DATA	Data	<AttributeData>	

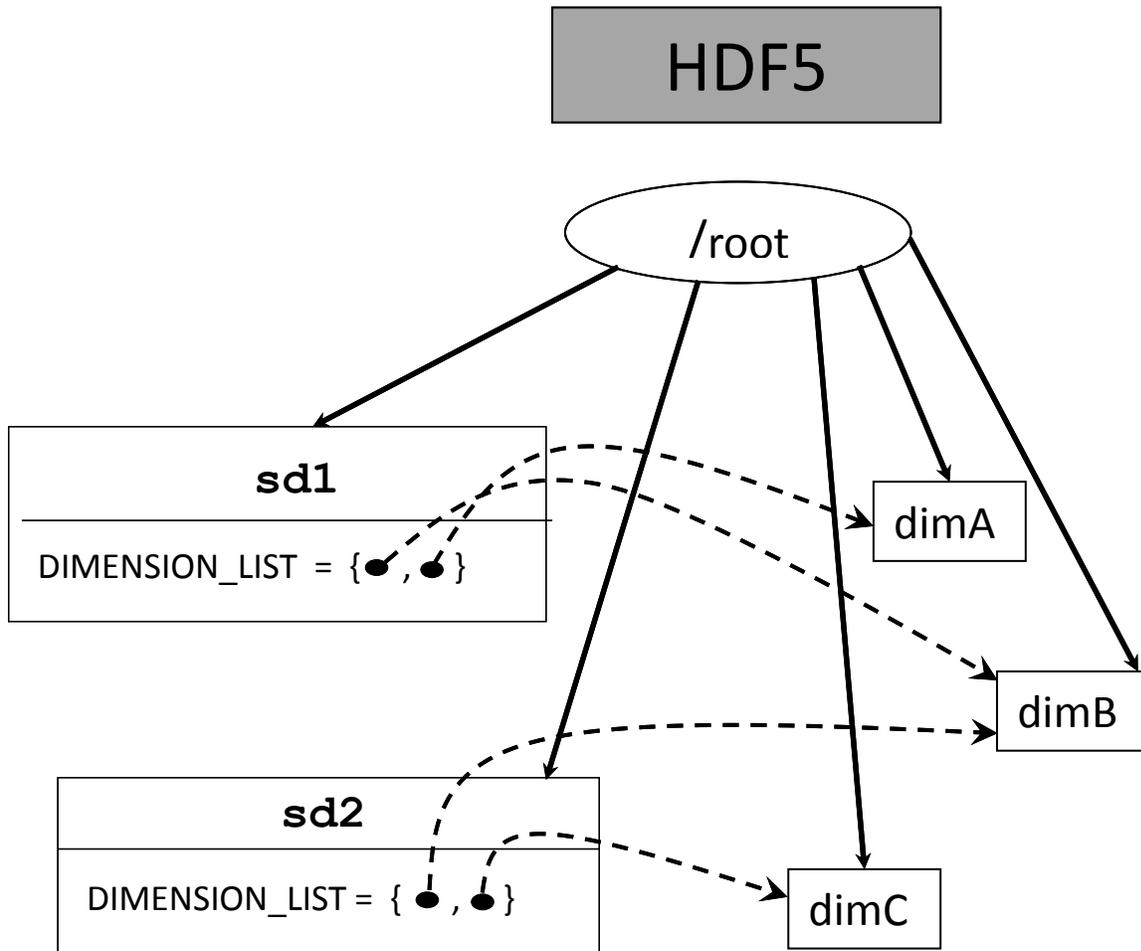
O	<SDS pre-defined attribute >			Same names, datatypes, etc., as the hdf4 counterpart
O	LONGNAME	Attr	.	
O	UNIT	Attr		
O	FORMAT	Attr		
O	COORDINATE_SYSTEM	Attr		
O	RANGE	Attr		
O	FILL_VALUE	Attr		The HDF5 Fill Value should be set to this value using the File Creation Property List.
O	SCALE_FACTOR	Attr		
O	SCALE_FACTOR_ERROR	Attr		
O	ADD_OFFSET	Attr		
O	ADD_OFFSET_ERROR	Attr		
O	packing_convention	Attr	Value should be "non-CF"	This attribute should be present if either SCALE_FACTOR or ADD_OFFSET are present.
O	Packing_convention_description	Attr	Value should be "unpacked = scale_factor x	This attribute should be present if either SCALE_FACTOR or ADD_OFFSET is present.

			(packed - add_offset)" if the HDF4 file follows the HDF4 packing rule documented at Section 3.10.6 of HDF4 user's guide [2].	
O	CALIBRATED_NT	Attr		

	<SDS Dimension with Name> (See Note 1)	Dataset		
R	NAME	Name	<name>	If a dimension doesn't have dimension scales, to achieve the interoperability with netCDF-4, the value of NAME should be NAME ="This is a netCDF dimension but not a netCDF variable."
R	SIZE	Dataspace		rank=1. Only the first dimension can be unlimited.
R	DATATYPE	Datatype	<HDF4 datatype>	
R*	DATA	Data	<value>	Required if a dimension has dimension scales.
O	<Dimension pre- defined attribute>			These are dimensions of an <SDS dimension with Name> dataset, not an SDS dataset
O	LONGNAME	Attr		
O	UNIT	Attr		
O	FORMAT	Attr		
O	<User-defined attribute>	Attr		Defined above.

Note 1. Dimension scales are unique to the HDF4 file—no two dimension scales can have the same name in HDF4. The dimension scale name is the unique key to distinguish this dimension from other dimensions or other HDF4 objects. So in HDF5, all of the dimension scales are stored under the root group as separate HDF5 dataset and it will not cause any ambiguity. Dimension names are <DimName1>, <DimName2>, ..., <DimNameN> as defined in HCR. Dimension scale datasets are identified in the HDF5 SDS dataset by the attribute `DIMENSION_LIST`. The attribute `DIMENSION_LIST` is an array of object references from the HDF5 SDS dataset to the dimension scale datasets. Figure 1 illustrates this structure. In the figure there are two 2-d datasets, `sd1` and `sd2`, each of which has two dimension scales. The datasets share the dimension scale `dimB`. Since netCDF-4 uses the HDF5 dimension scale APIs, part of the reason to follow the HDF5 dimension scale specification is to make the converted HDF5 netCDF-4 compatible.

Figure 1. How HDF4 Dimension Scales should be stored in HDF5.



Note 2. Global SDS attributes should be stored as attributes to the HDF5 root group. An SDS global attribute is to be stored in HDF5 as an attribute on the root group with the suffix “_GLO_SDS” or “_GLOSDS.” Other than these differences, an SDS global attribute is treated the same as a normal SDS attribute.

Table 3. SDS global attribute mapping

	<SDS global attribute>			Attribute assigned to the root group.
R	NAME	Attr name	<AttributeName>_GLO_SDS or <AttributeName>_GLOSDS	
R	DATATYPE	Datatype	<Attributetype>	
R	N_VALUES	Num-values	<AttributeCount>	
R	DATA	Data	<AttributeData>	

3.2 Vdata

Vdatas are mapped to 1 dimensional extendable HDF5 datasets of compound datatype. The HDF5 dataset should conform to the *HDF5 Table Specification* [5]. All attributes required by the specification should be included, even if not specified here.

Table 4. Vdata mapping

	Vdata	HDF5 object of component	Datatype, value, etc.	Notes
R	NAME	Attr	HDF4_OBJECT_NAME = <string> TITLE=<string>	<string> is the same as the corresponding Vdata name. TITLE is defined in the HDF5 Table Specification [5].
R	CLASS	Attr	CLASS= "TABLE"	The HDF5 CLASS must be "TABLE".
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. The fields of the HDF4 Vdata are members of an HDF5 compound datatype.
R	NAME	Field name	FIELD_(N)_NAME= <name>	The name of the HDF5 member type is <name>. See <i>HDF5 Table Specification</i> [5].
R	DATATYPE	member datatype	<HDF4 datatype>	The type of the compound element. Only Atomic types and one dimensional arrays.
R	ORDER	Num-values		If ORDER > 1, the HDF5 datatype should be an HDF5 ARRAY type.
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		

3.3 Vgroup

Vgroups are mapped individually to HDF5 groups. See section 4 for details on how to deal with the graph structures defined by the collection of Vgroups within an HDF4 file. See section 7 for how to deal with the HDF5 “root” group in an HDF4 file.

Table 5. Vgroup mapping.

	Vgroup	HDF5 object or component	Datatype, value, etc.	Notes
R	NAME	Attr	HDF4_OBJECT_NAME = <string>	<string> 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	HDF4_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.

3.4 Raster images

Raster images (8-bit, 24-bit, and general raster (GR)) are mapped to HDF5 datasets with simple 2D (8-bit) or 3D (24-bit or 8-bit) dataspace. Each element of the dataset is a scalar value. The HDF5 Image should conform to the *HDF5 Image and Palette Specification* [4]. All attributes required by the specification should be included, even if not specified here.

The HDF5 image conventions support additional information that is not supported in HDF4, such as image transparency and color models other than RGB. The general rule is that the HDF5 Image should describe the image as correctly as possible, even if this specification does not explicitly define the particular case

GR global attributes

A special case is the GR global attribute, which is an attribute that applies to all of the GR in a file. A GR global attribute is to be stored in HDF5 as an attribute on the root group with the suffix “GLO_GR” or “GLOGR.” Other than these differences, an GR global attribute is treated the same as a normal GR attribute.

Table 6. Raster image mapping.

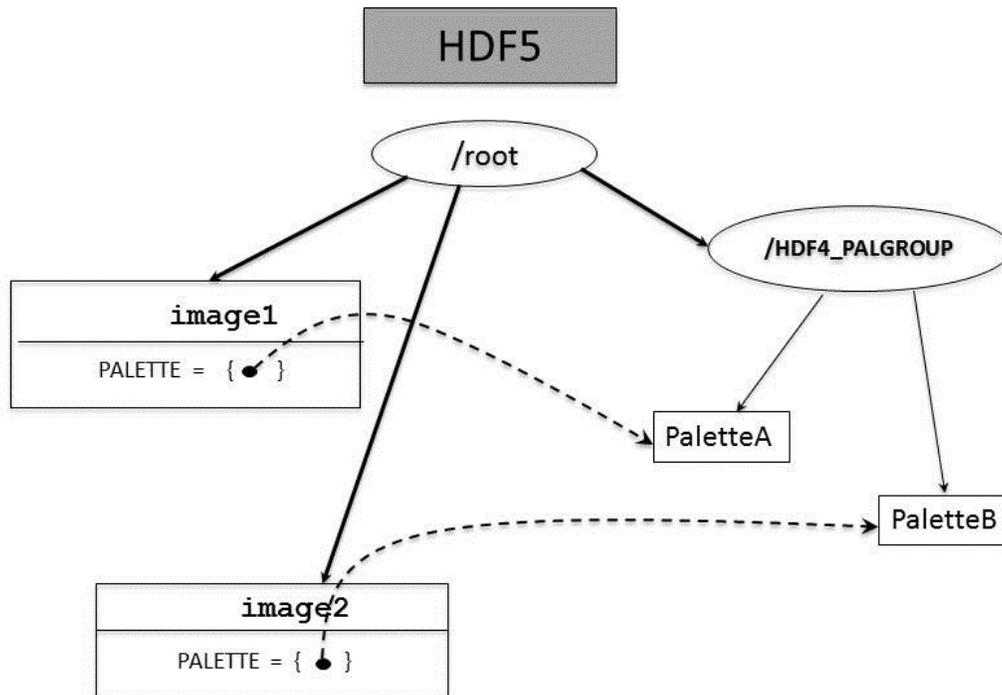
	Image	HDF5 object or component	Datatype, value, etc.	Notes
R	NAME	Attr	HDF4_OBJECT_NAME = <string>	<name> is the same as a GR name
R	(Pixel type)	Datatype		Atomic numeric type.
R	N_COMPS	Num values	<size of component dimension>	
R	COMP_TYPE	Atomic type	<HDF4 datatype>	
R	DIMENSIONSIZE	Dataspace		rank=2 if N_COMPS is 1. rank =3 if N_COMPS is > 1, one of the dimensions is the pixel components.
R	(image array)	Data		
O	<User-defined attribute>	Attr	<name> = <value>	rank = 1; size is fixed;
R	(Class)	Attr	CLASS = "IMAGE"	Required by HDF5 image spec.
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE = "raster8", "raster24" or "GR"	
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
R ²	<Image palette>		PALETTE = {object_ref1, object_ref2, ... object_refn}	PALETTE is an array of object references that refers to the corresponding palettes. See notes 1 and 2.
O ²	(palette namelist)	Attr	PALETTE_NAMELIST = {palette_name1, palette_name2, ...}	For version 3 and earlier of this document, this attribute is required. Since the HDF5 image and palette specification doesn't require this attribute. So we change this as an optional attribute.
R ²	(Image subclass)	Attr	IMAGE_SUBCLASS = "IMAGE_INDEXED" or "IMAGE_TRUECOLOR"	If N_COMPS == 1, "IMAGE_INDEXED" If N_COMPS == 3, "IMAGE_TRUE_COLOR". Any other image is undefined in this specification.
R ^{2,3}	(Color model)	Attr	IMAGE_COLORMODEL = "RGB"	The IMAGE_COLORMODEL should be the same as the PALETTE_COLORMODEL of its palettes. See note 3.
R	INTERLACEMODE	Attr	INTERLACE_MODE = "INTERLACE_PIXEL" or "INTERLACE_PLANE"	The HDF5 dataset should use the same interlace mode as the HDF4 image. If the HDF4 image used "MFGR_INTERLACE_LINE", the HDF5 Image should use INTERLACE_PIXEL.
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 1. HDF5 image and palette specification allows more than one palette in PALETTE, but in a translation from HDF4 to HDF5 there will be only one palette and palettes are stored in a special group called "/HDF4_PALGROUP". Figure 2 illustrates this structure. In the figure there are two images, image1 and image2, each of which has an attached palette.

Note 2. These are required if there is a palette associated with the image.

Note 3. In principle, an HDF4 image could have any number of components and use any color model, but the HDF4 standard only defines 8-bit and 24-bit RGB images. With no other information, the HDF5 image is assumed to use an RGB color model. The general rule is to convert the HDF4 image into an HDF5 image that stores its properties. Therefore, if the color model of the HDF4 image is known, the HDF5 image should be defined to use that color model, whatever it may be.

Figure 2. How HDF4 palettes should be stored in HDF5 and referenced by images.



3.5 Palette

Palettes are mapped to HDF5 datasets that are 2-D arrays of bytes with dimensions 256 x 3. The HDF5 Palette should conform to the *HDF5 Image and Palette Specification* [4]. All attributes required by the Palette Specification should be included, even if not specified here. Notice that the HDF5 palette conventions support additional information that is not supported in HDF4, such as color model and range index. In the event that the HDF4 file contains an object that is not an HDF4 palette but should be an HDF5 Palette (e.g., a table that represents an range index color table), it should be stored as an HDF5 Palette if possible.

As indicated in the “Image” section, it is recommended that in HDF5 all palette objects from an HDF4 file be stored in the group “/HDF4_PALGROUP/<palette_name>.” The HDF5 link <palette_name> can be given any name, but if no name is available <palette_name> should be assigned according to the guidelines in section 4.

Table 7. Palette mapping.

	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 = 256 x 3 (See note 1.)
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE= “palette”	
O	<User-defined attribute>	Attr	<name> = <value>	rank = 1; size is fixed;
O	(Reference number)	Attr	HDF4_REF_NUM = <uint16>	
R	(Class)	Attr	CLASS = “PALETTE”	Required by the palette specification.
R	(Palette type)	Attr	PAL_TYPE = “STANDARD8”	
R	(Color model)	Attr	PAL_COLORMODEL = “RGB”	

Note 1: 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).

3.6 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 (but see Section 6.2 and 7.1 for more information about the treatment 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: HDF4_FILE_LABEL_<n>, HDF4_FILE_DESCRIPTION_<n>, HDF4_OBJECT_LABEL_<n>, HDF4_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 HDF4_OBJECT_LABEL_1 and HDF4_OBJECT_LABEL_2. Table 8 summarizes these guidelines.

Table 8. Annotation mapping.

	Annotation	HDF5 object or component	Datatype, value, etc.	Notes
R	(File label)	Attr	HDF4_FILE_LABEL_<n> = <string>	Root group attribute.
R	(File description)	Attr	HDF4_FILE_DESCRIPTION_<n> = <string>	Root group attribute.
R	(Object label)	Attr	HDF4_OBJECT_LABEL_<n> = <string>	Attribute for corresponding HDF5 object. (See note 1.)
R	(Object description)	Attr	HDF4_OBJECT_DESCRIPTION_<n> = <string>	Attribute for corresponding HDF5 object. (See note 1.)

Note 1. Although any HDF4 object that can be represented by a tag can have associated object labels and descriptions, this specification only requires that annotations be converted for the following tags:

1. SDS-related tags DFTAG_SD, DFTAG_SDG, DFTAG_NDG. Annotations on these objects will be mapped to an HDF5 attribute called HDF4_SDS_LABEL and HDF4_SDS_DESC.
2. Image-related tags DFTAG_RI, DFTAG_RI8, DFTAG_RIG. Annotations on these objects will be mapped to an HDF5 attribute called HDF4_IMAGE_LABEL and HDF4_IMAGE_DESC.
3. Vgroup tag DFTAG_VG. Annotations on these objects will be mapped to an HDF5 attribute called HDF4_VGROUP_LABEL and HDF4_VGROUP_DESC.
4. Vdata-related tags DFTAG_VH, DFTAG_VS. Annotations on these objects will be mapped to an HDF5 attribute called HDF4_VDATA_LABEL and HDF4_VDATA_DESC.
5. Palette tag DFTAG_LUT. Annotations on these objects will be mapped to an HDF5 attribute called HDF4_PAL_LABEL and HDF4_PAL_DESC.

4 HDF4 and HDF5 File Organization

In addition to mapping individual objects, it is necessary to map the organization and name structure of the HDF4 file to the HDF5 file. The following guidelines are covered in this section:

1. Object names should be translated in a standard way. Objects that have no name in HDF4 (such as palettes) must be assigned a default name in the HDF5 file. If two or more HDF4 objects have the same name, they must be assigned unique names in HDF5.
2. The hierarchical structure of the HDF4 file should be accurately reproduced in the HDF5 file, but objects used internally by the HDF4 library (e.g., “CDF0.0”) should *not* be represented in the HDF5 file.
3. Shared objects (such as Named Dimensions) should be stored once and shared in the HDF5 file.

4.1 Object Names

The treatment of object names in HDF 4 (and earlier versions) varies depending on the type of object. The HDF4 User’s Guide provides details on naming HDF objects. In summary:

- Palettes and annotations do not have names.
- Vgroup and Vdata names are optional.
- Images may or may not be assigned names, depending on the version of the library, and in any case are optional.
- SDS objects are treated differently by different versions of the library. Early versions of the HDF library (before HDF 3.2) did not require an SDS to have a name, and they allowed two or more SDSs to have the same name. In later versions every SDS was required to have a unique name, and in addition it was possible to assign a second name, called LONGNAME that did not have to be unique.

Hence, in HDF4, names may occur in a variety of ways:

- An HDF4 object may have a unique name
- An HDF4 object may have no name
- More than one HDF4 object may have the same name

For the purposes of this specification, we can think of the name of an object in HDF4 (if one exists) as a kind of object attribute. It may not be stored specifically as an attribute, but it nevertheless plays a similar role. We refer to this as NAME in this paper.

In HDF5, the name of an object is the final component of its path within the file. For instance, if an HDF5 object has the path `/foo/bar`, its HDF5 name is “bar.”

4.1.1 Objects with unique names

In general, if an HDF4 object has a unique NAME attribute, the corresponding HDF5 object’s path should use that name. In addition, as the tables in Section 3 indicate, that same name can be stored as an attribute of the HDF5 object with the name “HDF4_OBJECT_NAME.”

4.1.2 Objects with no name

When an HDF4 object with no name must be stored in HDF5, it can be assigned either a default name, or a name provided by an application. A default name should be constructed from a default string with the object’s reference number concatenated on the end. For instance, an unnamed Vgroup whose reference number is 2 would be named `HDF4_VGROUP_2`. Table 9 shows templates for the default HDF5 names for HDF4 objects.

Table 9. Default HDF5 names for HDF4 objects.

HDF4 Object Type	HDF5 Object Name
Vgroup	HDF4_VGROUP_<ref>
Vdata	HDF4_VDATA_<ref>
SDS	HDF4_SDS_<ref>
Image	HDF4_IMAGE_<ref>
Palette	HDF4_PALETTE_<ref>

4.1.3 Name Collisions

When two HDF4 objects have the same name and will be written to the same HDF5 group, it is necessary to change the pathname of one or both of the objects. For instance, suppose two HDF4 SDSs have the name “mouse”, and both belong to the top-level Vgroup “cat.” Using the rule suggested in section 4.1.1, these would both be stored as HDF5 datasets with the path “/cat/mouse.” Since no two objects in HDF5 can have the same path, it is necessary to change the pathname of one or both of these datasets. The following rules are used for dealing with this situation.

When two HDF4 objects have names that map to the same HDF5 name according to this specification (a “collision”), applications are free to change the names to anything they like. If no name is assigned, then all but one of the colliding objects in the HDF5 file should be given a default name, using the templates in Table 9.

Notice that the original non-unique HDF4 name can still be stored in the attribute `HDF4_OBJECT_NAME` associated with the corresponding HDF5 object.

4.2 Organization of the HDF4 and HDF5 Files

The grouping structure of an HDF4 file should be represented in the corresponding HDF5 file, but with some exceptions, as described here.

In general, a user-defined HDF4 Vgroup should be mapped to an HDF5 Group with a corresponding path name. Lone Vgroups should be children of the HDF5 root group.

An HDF4 file can consist of a set of individual objects, but it may also be organized as a directed graph, with Vgroups as nodes and any HDF4 objects as nodes. Any HDF4 object may be a child of no Vgroup, one Vgroup, or more than one Vgroup. Since HDF5 can also be organized as a directed graph (using the HDF5 group structure), a mapping from HDF4 to HDF5 should replicate the graph structure as much as possible. Hence, as a general rule, *HDF4 objects that are members of a Vgroup are stored as children of a corresponding HDF5 Group.*

One difference between HDF4 and HDF5 is that there is always at least one group in HDF5, the root group, and all objects must be descendants of the root group. For purposes of the HDF4-to-HDF5 mapping, the root group in HDF5 can be thought of as the same as an HDF4 file. Hence, *HDF4 objects that are not members of a Vgroup should be stored as children of the root group in the HDF5 file.*

A special case of the HDF4 grouping structure that requires attention is a cyclic structure, where the links between groups form a cycle. In this case there is no “top” group, and hence no obvious rule for which group should be assigned to the HDF5 root group. The rule proposed for handling this situation is to arbitrarily choose any vgroup in the ring and place it under root group. That is, *when Vgroups form a cycle, the cycle of groups should be converted into HDF5 by arbitrarily choosing any vgroup in the cycle and placing it under the root group.*

Although the grouping structure of an HDF5 file reflects that intended by an application, an HDF4 file that contains SDS or GR objects is likely to contain extra Vgroups that were not defined by the application. These are special Vgroups and other structures used by the HDF library to organize the SDS and GR objects. These extra objects are hidden from the SDS and GR API, and are used only to help describe the internal organization of the HDF4 file. For example, Vgroups of class “CDF0.0” and “RI0.0” contain all of the SDSs and GRs in an HDF file, together with global SDS attributes and global GR attributes, respectively. *Since these internal objects are not needed to describe HDF4 objects in HDF5, they should be ignored in the conversion process.*

Figure 3 provides examples of how some of these structures occur in an HDF4 file:

- The normal Vgroups VgroupA and VGBa2 both contain the dataset SDS1. That relationship is replicated in the HDF5 file.
- The Vgroups CDF0.0 and RI0.0 are written by the HDF4 library, as a way of organizing the HDF4 file. If an application accesses the file using the SD or GR interfaces, the application never sees these two groups, and consequently they should be ignored in any conversion to HDF5.
- In Figure 3 there is a Vgroup cycle between VgroupB and VGBa. When converting to HDF5, one of the groups in the cycle is attached to the root, in this case VgroupB

Figure 4 shows how the file in Figure 3 would look in HDF5 using these conventions..

Figure 3. HDF4 example, showing special internal groups (CDF0.0 and RI0.0), and cyclic grouping structure

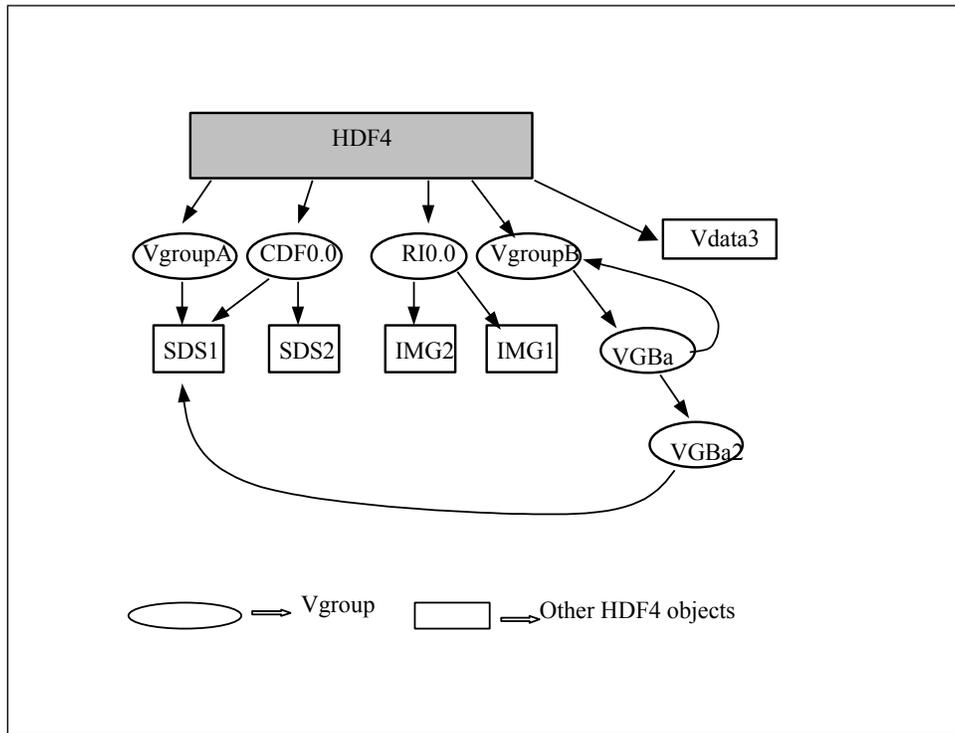
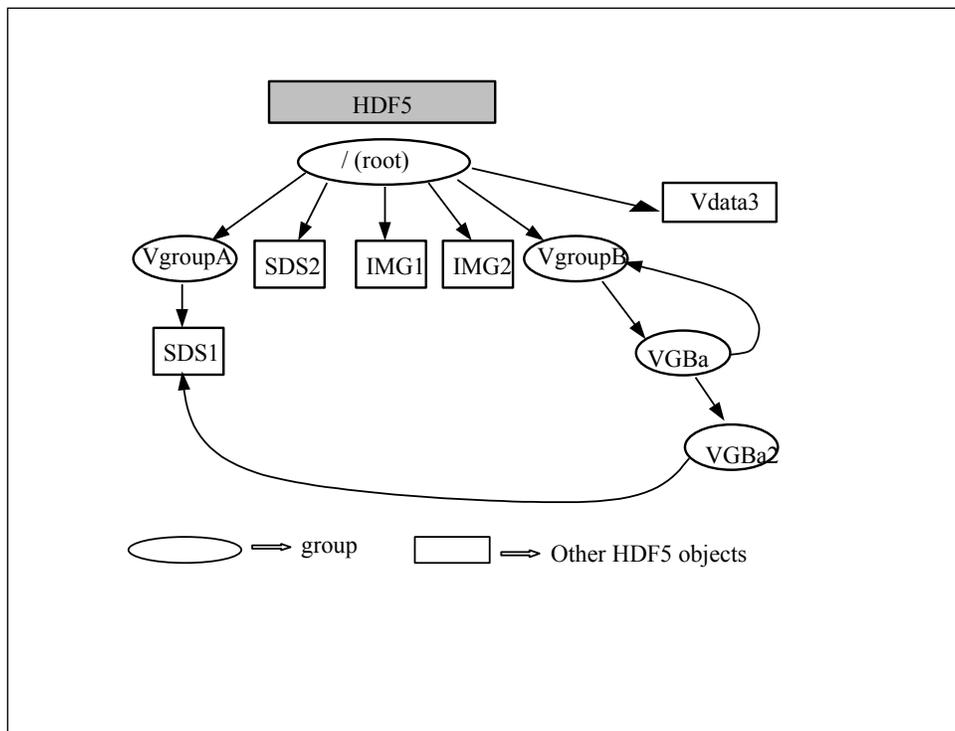


Figure 4. Illustrative Example of HDF5 Structure



4.3 Shared HDF4 objects: dimensions and palettes

Dimension scales (coordinate variables) and palettes can be shared in HDF4. The following rules apply to storing these objects in HDF5.

- By following the HDF5 dimension scale specification, the HDF4 Dimension Scales are stored under the HDF5 root group. This also makes the HDF5 files interoperable with netCDF-4. The HDF4 Dimension Scales are globally unique to the HDF4 file, so one copy of each object is stored.
- An extra group “HDF4_PALGROUP” is created under the HDF5 root group. This group is used to store the HDF5 datasets constructed for HDF4 Palettes. . The palettes are globally unique to the HDF4 file, so one copy of each object is stored under this group.
- Attributes that are global attributes of the SDS and GR interfaces are stored as attributes of the HDF5 root group.

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 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 “HDF4_REF_NUM” whose type is unsigned 16-bit integer 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.

5.2 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.
- The HDF5 file offset (address width) and length (max address) should be ignored.
- Other HDF5 file properties should be ignored.

5.3 Storage properties

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 the HDF5 dataset can be stored without applying that property.

5.4 Dataspaces

The HDF5 “dataspace” concept generalizes and formalizes the ideas of dimensionality that are implicit in HDF4 objects. Determining the appropriate dataspace of the HDF5 object is straightforward. This mapping was given for each object in Section 2, and is summarized in Table 10.

Table 10. Summary of HDF5 dataspace for HDF4 objects.

HDF4 Object	HDF5 Dataset	Notes
SDS: rank, dimensions	Simple dataspace, rank, dimensions. maxdims[0] = H5S_UNLIMITED or maxdims[0] = dimsizes[0]	The first dimension of the HDF4 SDS may be UNLIMITED. In this case, the first dimension of the HDF5 dataspace would then have maxdims = H5S_UNLIMITED
GR, RI8, RI24	Simple dataspace, rank = 2 or 3, dimensions = dimsizes, third dimension is the number of components per pixel.	Datatype is 8-bit integer. The components of 24-bit images are one of the dimensions of a 3-dimensional image. See [4].
Palette	Simple dataspace, rank = 2, dimensions= {256, 3}	See [4].
Vdata	Simple dataspace, rank = 1, dimension = H5S_UNLIMITED or dimension = <maximum records>	Datatype is compound. See [5].

5.5 Datatypes

When converting SDSs, Vdatas, and images from HDF4 files to HDF5 files, the HDF4 data elements need to be converted to appropriate corresponding HDF5 data types. The following rules describe how HDF4 datatypes should be converted to HDF5:

- 1) **“Standard” numeric datatypes.** HDF4 standard integer and floating point datatypes should be converted to corresponding HDF5 datatypes. The size, type (integer vs. float), “endianness,” etc., should be maintained. Table 11 describes the correspondence between standard HDF4 type definitions and HDF5 type definitions.

Table 11. Correspondence between standard numeric datatypes.

HDF4 type	Corresponding HDF5 type
DFNT_INT8	H5T_STD_I8BE
DFNT_UINT8	H5T_STD_U8BE
DFNT_LINT8	H5T_STD_I8LE
DFNT_LUINT8	H5T_STD_U8LE
DFNT_INT16	H5T_STD_I16BE
DFNT_UINT16	H5T_STD_U16BE
DFNT_LINT16	H5T_STD_I16LE
DFNT_LUINT16	H5T_STD_U16LE
DFNT_INT32	H5T_STD_I32BE
DFNT_UINT32	H5T_STD_U32BE
DFNT_LINT32	H5T_STD_I32LE
DFNT_LUINT32	H5T_STD_U32LE
DFNT_FLOAT32	H5T_IEEE_F32BE
DFNT_LFLOAT32	H5T_IEEE_F32LE
DFNT_FLOAT64	H5T_IEEE_F64BE
DFNT_LFLOAT64	H5T_IEEE_F64LE

Applications may choose to use any equivalent HDF5 data type, e.g., H5T_STD_I32LE instead of big endian.

Note that the HDF5 memory data type should always be a native type, H5T_NATIVE_*

- 2) **“Native” datatypes (e.g. DFNT_NINT16).** Because HDF4 files contain no information about the actual architecture that a native type was written from, it may not be possible for an application to decipher the value of an HDF4 native type for conversion. Therefore, HDF4 native format types should be treated with caution. If a converter can determine the value of a native type, it should choose a corresponding HDF5 type and perform the conversion. If it can not, the proper strategy is left to the discretion of the application.
- 3) **Character datatype.** For the HDF4 datatypes DFNT_CHAR8 and DFNT_UCHAR8, there is no corresponding datatype in HDF5. If the values in an HDF4 SDS, raster image, palette or dimensional scale data of an SDS are of one of these two types, it is recommended that they be converted to H5T_STD_I8BE or H5T_STD_U8BE. If the values of a Vdata field or attribute are of one of these types, it is recommended that those values be interpreted as constituting a string, and that they be converted to H5T_STRING.

6 Interpreting HDF5 files as HDF4 files when there is no explicit metadata

Sections 3-5 deal with how to represent HDF4 objects in HDF5. If these guidelines are followed in creating HDF5 files, it should be possible for an “HDF4 application” to make reasonable use of data stored in HDF5. But what

happens if an HDF4 application encounters an HDF5 file that does not contain the extra information specified in these guidelines? This section provides guidelines on how an HDF4 application can interpret a “raw” HDF5 file.

If an HDF4-based application encounters an object in an HDF5 file that does not contain the metadata described in the previous sections, three possible outcomes can occur:

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

6.1 Case 1: No HDF4 counterpart

Only HDF5 datasets, attributes and groups can have HDF4 counterparts. All other HDF5 objects should be assumed to have no HDF4 counterpart. For instance, the HDF5 “named datatype” object has no HDF4 counterpart.

Certain HDF5 datasets, attributes and groups also have no HDF4 counterpart. These include

- 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 can map to Vdatas.
- Any HDF5 object whose size is greater than $2^{31}-1$ bytes.
- Any HDF5 attribute of an HDF5 Image (other than those specified in this document)
- Any HDF5 link that does not point to a corresponding HDF4-compatible object.

6.2 Case 2: The ambiguous case

There are a number of cases where an HDF5 object could be interpreted as any of a number of HDF4 objects. Here are some such cases.

6.2.1 HDF5 “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 dataset with an HDF4-compatible scalar datatype is assumed to be an HDF4 SDS.

Note that HDF5 Datasets conforming to the *HDF5 Image and Palette Specification* [4] or *HDF5 Table Specification* [5] will have metadata that unambiguously indicate that the dataset should be mapped to a GR, Palette, or Vdata, respectively.

6.2.2 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 the convention:

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

A special case is a file that has been converted from HDF4 to HDF5. In this case, the root group should be ignored. See section 7.

6.2.3 Strings

Since strings are not explicitly supported in HDF4, it is recommended that any HDF5 string be mapped into an array of characters in HDF4. Thus, for example, an HDF5 one-dimensional array of type H5T_STRING should be translated to an HDF4 two-dimensional array of type ‘char.’

HDF5 strings may be null-terminated, or may have padding (0-padding or space-padding), and may be variable length. In general, a corresponding HDF4 array must be fixed length, and should be terminated or padded in the equivalent fashion, if the HDF4 application can handle it. Otherwise, the treatment of an HDF5 string is left to the application.

6.3 Case 3: The mapping is unambiguous

Any HDF5 dataset that is not covered in case 1 or case 2 should map to an HDF4 SDS or Vdata. HDF5 groups map to Vgroups, with the exception of the root group, covered above.

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 HDF4 annotation for the corresponding HDF4 object.

HDF5 Table datasets should be mapped to Vdatas.

HDF5 Image datasets should be mapped to GR images.

HDF5 Palette datasets should be mapped to HDF4 palettes associated with GR images.

7 Special HDF5 Objects created by this mapping.

This section discusses special cases that arise in an HDF5 file that was created using the HDF4 to HDF5 mapping: the HDF5 root group, and HDF5 objects created by the HDF4 to HDF5 mapping. Earlier sections give detailed mappings for the objects: HDF5 Groups are converted to Vgroups, HDF5 Datasets to SDS or other objects, and attributes are converted to HDF4 attributes or annotations.

The general principle is:

Try to create an HDF4 file that faithfully reproduces the HDF5 file

7.1 Converting the Root Group

In many cases, the HDF4 file should not have a single 'root' Vgroup. In these cases, it is recommended that the HDF5 root group should be omitted from the HDF4 file. If the root group is omitted, the attributes of the HDF5 root group should be stored in the HDF4 file as File Description annotations. The value of the annotation should be converted to a string, of the form "HDF5_<attributename>=<attribute value>".

7.2 Objects and attributes created by the HDF4 to HDF5 mapping

The document specifies special HDF5 objects that represent HDF4 objects and metadata. When converting from HDF5 to HDF4, these objects should be converted to the appropriate HDF4, to recreate the original HDF4 objects. In some cases, the objects should be omitted from the HDF5.

- 1) Attributes that cannot be recreated in the HDF4 file (such as HDF4_REF_NUM) should be ignored.
- 2) Special attributes, such as HDF4_OBJECT_NAME, etc., should be used to create the corresponding HDF4 attribute or metadata. Attributes that represent global SDS or GR annotations (the HDF5 attribute name ends in "_GLO_SDS", "_GLOSDS", "_GLO_GR", or "_GLOGR"), should be converted to SD or GR annotations respectively.
- 3) The special groups /HDF4_PALGROUP should be omitted from the HDF4 file.
- 4) For an HDF5 Image, the PALETTE attribute should be ignored. If the PALETTE_NAMELIST is present, it should be used to create an appropriate HDF4 palette from the HDF5 Palette objects. If the PALETTE_NAMELIST is not present, applications can use the HDF5 APIs to retrieve the palette names based on the values stored in the PALETTE attribute.
- 5) For an HDF5 SDS, applications can use the HDF5 APIs to retrieve the dimension scales based on the values stored in the DIMENSION_LIST attribute.

7.3 Datatypes

When converting HDF5 datasets to HDF4 SDSs, Vdatas, and images, the HDF5 data elements need to be converted to appropriate corresponding HDF4 data types. For Atomic datatypes the rules given in Section 5.5 can be applied.

- 1) **“Standard” numeric datatypes.** Atomic HDF5 datatypes should be converted to corresponding HDF4 standard integer and floating point datatypes, as given in Table 11. Applications may choose to use any equivalent for the HDF5 data type, e.g., `H5T_STD_I32LE` might be mapped to `DFNT_INT32` or `DFNT_LINT32`.

Note that the HDF5 memory data type should always be a native type, `H5T_NATIVE_*`.

- 2) **String datatype.** HDF5 `STRING` datatype should be converted to array of `DFNT_CHAR8`.
- 3) **Other Atomic datatypes.** Some HDF5 datatypes, such as `ENUM`, `bitfield`, `opaque`, and similar cases, may be mapped to integers or character arrays in the HDF4 object. This document does not specify how to convert these datatypes, applications are free to define their own.
- 4) **Other datatypes.** Some datatypes, such as variable length strings, nested compound datatypes, and similar cases, may be mapped to “flattened” datatypes with fixed length. This document does not specify how to convert these datatypes, applications are free to define their own.

Appendix A: How SDS dimension is mapped in the version 3

Since the versions prior to 2.2.3 of the HDF4 to HDF5 conversion toolkit were implemented according to the SDS dimension mapping documented in the version 3 of this document, we still listed the information here on how HDF4 SDS dimension is mapped to HDF5 in the version 3. This can be a good reference to the older version conversion toolkit users. Starting from the version 2.2.3 of the conversion toolkit, the version 3 mapping is still kept as an optional feature.

Table 12. SDS mapping

	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		(See note 1)
O	(HDF4 object type)	Attr	HDF4_OBJECT_TYPE = "SDS"	

	<SDSArray>			
R	NAME	Attr	HDF4_OBJECT_NAME = <SDSArrayName>	See Section 4 for details on how NAME is used as a link in HDF5 .
R	DATATYPE	Datatype	<HDF4 datatype>	
R	DIMENSIONRANK & DIMENSIONSIZE	Dataspace		Dimension sizes are also part of dimension information.
R	DIMENSIONLIST	Attr	DIMENSIONLIST = {object_ref1, object_ref2, ... object_refn}	An array of object references that refer to the corresponding. dimension datasets. See note 1.
R	DIMENSION_NAMELIST	Attr	DIMENSION_NAMELIST = {<DimName1, <DimName2>, ..., <DimNameN>}	The absolute paths of dimensions are stored. Dimension names are defined in the HCR specification. See note 1.
R	(Data)	Data		See section 0 for details on how to handle datatypes.
O	<User-defined attribute >	Attr		rank = 1; size is fixed. Global attributes: see note 2.
O	<SDS pre-defined attribute >	Attr		
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		rank = 1; size is fixed. Global attributes are covered below.
R	NAME	Attr name	<AttributeName>	
R	DATATYPE	Datatype	<AttributeType>	
R	N_VALUES	Num-values	<AttributeCount>	
R	DATA	Data	<AttributeData>	

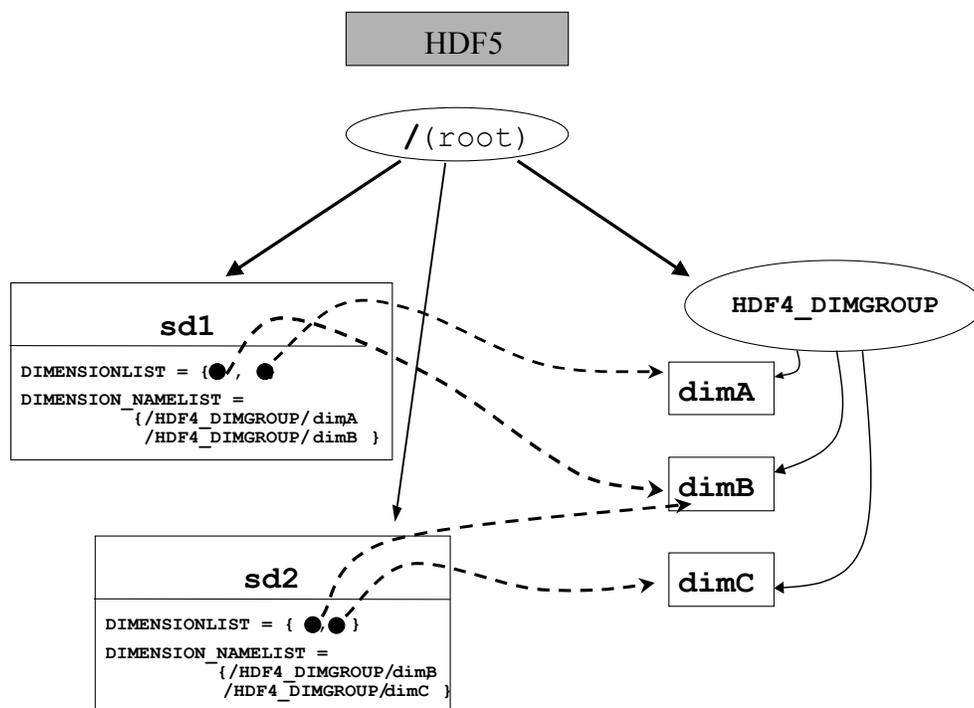
O	<SDS pre-defined attribute >			Same names, datatypes, etc., as the hdf4 counterpart
---	------------------------------	--	--	--

O	LONGNAME	Attr	.	
O	UNIT	Attr		
O	FORMAT	Attr		
O	COORDINATE_SYSTEM	Attr		
O	RANGE	Attr		
O	FILL_VALUE	Attr		The HDF5 Fill Value should be set to this value using the File Creation Property List.
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> (See Note 1)	Dataset		
R	NAME	Name	<name>	See note 1.
R	SIZE	Dataspace		rank=1. Only the first dimension can be unlimited.
R	DATATYPE	Datatype	<HDF4 datatype>	
R	DATA	Data	<value>*	
O	<Dimension pre-defined attribute>			These are dimensions of an <SDS dimension with Name> dataset, not an SDS dataset
O	LONGNAME	Attr		
O	UNIT	Attr		
O	FORMAT	Attr		
O	<User-defined attribute>	Attr		Defined above.

Note 1. Dimension scales are unique to the HDF4 file—no two dimension scales can have the same name in HDF4. In HDF5, all of the dimension scales are stored in the group called /HDF4_DIMGROUP. Dimension names are <DimName1>, <DimName2>, ..., <DimNameN> as defined in HCR. Dimension scale datasets are identified in the HDF5 SDS dataset in two ways: by the attribute DIMENSIONLIST, and by the attribute DIMENSION_NAMELIST. The attribute DIMENSIONLIST is an array of object references to the dimension scale datasets DIMENSION_NAMELIST is an array containing the names of the dimension scale datasets. Figure 1 illustrates this structure. In the figure there are two 2-d datasets, sd1 and sd2, each of which has two dimension scales. The datasets share the dimension scale dimB.

Figure 4. How HDF4 Dimension Scales were stored in HDF5 in version 3 of the mapping document.



References

1. HCR Documentation: https://support.hdfgroup.org/products/hdf4_tools/hcr.html
2. HDF4 Documentation. <https://support.hdfgroup.org/release4/doc/index.html>
3. HDF5 Documentation. <https://support.hdfgroup.org/HDF5/doc/index.html>
4. HDF5 Image and Palette Specification. <https://support.hdfgroup.org/HDF5/doc/ADGuide/ImageSpec.html>
5. HDF5 Table Specification. https://support.hdfgroup.org/HDF5/doc/HL/H5TB_Spec.html
6. HDF5 Dimension Scale Specification. https://support.hdfgroup.org/HDF5/doc/HL/H5DS_Spec.pdf
7. HDF5 for HDF4 Users: a short guide, <https://gamma.hdfgroup.org/papers/papers/h4toh5/HDF5forHDF4Users.pdf>