* Sets and Groups

# Chapter Overview

This chapter discusses the roles of the following sets and groups in organizing data stored in an HDF file:

Raster image sets (RIS) Raster image groups (RIG)

Scientific data sets (SDS) Scientific data groups (SDG)  
 Numeric data groups (NDG)  
 SDG-like NDGs

Vsets Vgroups and Vdatas

Raster-8 sets (obsolete)

This chapter introduces several tags used in support of sets and groups. All of these tags are fully described in Chapter , Tag Specifications, and are listed in the table in Appendix A, Tags and Extended Tag Labels.

## Data Sets

HDF files frequently contain several closely related data objects. Taken together, these objects form a data set which serves a particular user requirement. For example, five or six data objects might be used to describe a raster image; eight or more data objects might be used to describe the results of a scientific experiment.

The HDF mechanism for specifying and controlling data sets is the group. The data element of a group consists of a single record listing the tag/refs for all the objects contained in the data set. For example, the raster image groups described in the following sections each contain three tag/refs that point to three data objects that, taken as a set, fully describe an 8-bit raster image.

### Types of Sets

The current HDF implementation supports three kinds of sets:

Raster image set A set containing a raster image and descriptive information such as the image dimensions and an optional color lookup table

Scientific data set A set containing a multidimensional array and information describing the data in the array

Vset A general grouping structure containing any kinds of HDF objects that a user wishes to include

Each HDF set is defined with a minimum collection of data objects that will make sense when the set is used. For example, every raster image set must contain at least the following data objects:

Raster image group The list of the members of the set

Image dimension record The width, height, and pixel size of the raster image

Raster image data The pixel values that make up the image

In addition to the required objects, a set may include optional data objects. An 8-bit raster image set, for instance, often contains a palette, or color lookup table, which defines the red, green, and blue values associated with each pixel in the raster image.

### Calling Interfaces for Sets

The HDF Group provides calling interfaces for all the HDF sets that it supports. These interfaces provide routines for reading and writing the data associated with each set. The currently supported HDF libraries are callable from either C or FORTRAN programs.

In addition to the libraries, a growing number of command-line utilities are available to manipulate sets. For example, a utility called r8tohdf converts one or more raw raster images to HDF 8-bit raster image set format.

The calling interfaces are described in the document NCSA HDF Calling Interfaces and Utilities for Versions 3.2 and earlier and in the HDF User's Guide and HDF Reference Manual for Versions 3.3 and 4.x.

## Groups

As discussed above, HDF data objects are frequently associated as sets. But without some explicit identifying mechanism, there is often no way to tie them together. To address this problem, HDF provides a grouping mechanism called a group. A group is a data object that explicitly identifies all of the data objects in a set.

Since a group is just another type of data object, its structure is like that of any other data object; it includes a DD and a data element. But instead of containing the pixel values for a raster image or the dimensions of an array, a group data element contains a list of tag/refs for the data objects that make up the corresponding set.

A group tag can be defined for any set. For instance, the raster image group tag (DFTAG\_RIG) is used to identify members of raster image sets; the RIG data element lists the tag/refs for a particular raster image set.

An Example

Suppose that the two images shown in Figure 2e, "Physical Representation of Data Objects," are organized into two sets with group tags. Since they are raster images, they may be stored as RIGs. Figure 5a, "Physical Organization of Sample RIG Groupings," illustrates the use of RIGs with these images.

* Physical Organization of Sample RIG Groupings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Offset | Item | Contents |  |  |  |
| 0 | FH | 0e031301 |  |  | (HDF magic number) |
| 4 | DDH | 10 0L |  |  |  |
| 10 | DD | DFTAG\_FID | 1 | 130 | 4 |
| 22 | DD | DFTAG\_FD | 1 | 134 | 41 |
| 34 | DD | DFTAG\_LUT | 1 | 175 | 768 |
| 46 | DD | DFTAG\_ID | 1 | 943 | 4 |
| 58 | DD | DFTAG\_RI | 1 | 947 | 240000 |
| 70 | DD | DFTAG\_ID | 2 | 240947 | 4 |
| 82 | DD | DFTAG\_RI | 2 | 240951 | 240000 |
| 94 | DD | DFTAG\_RIG | 1 | 480951 | 12 |
| 106 | DD | DFTAG\_RIG | 2 | 480963 | 12 |
| 118 | DD | DFTAG\_NULL |  | *(Empty)* | |
| 130 | Data | sw3 |  |  | |
| 134 | Data | solar wind simulation: third try. 8/8/88 | | | |
| 175 | Data | . . . |  | *(Data for image palette)* | |
| 943 | Data | 400, 600 ... |  | *(Data for 1st image dimension record)* | |
| 947 | Data | ... |  | *(Data for 1st raster image)* | |
| 240947 | Data | 400, 600 ... |  | *(Data for 2nd image dimension record)* | |
| 240951 | Data | ... |  | *(Data for 2nd raster image)* | |
| 480951 | Data | DFTAG\_IP8/1, DFTAG\_ID/1, DFTAG\_RI/1 | | | |
|  |  |  |  | *(Tag/refs for 1st RIG)* | |
| 480963 | Data | DFTAG\_IP8/1, DFTAG\_ID/2, DFTAG\_RI/2 | | | |
|  |  |  |  | *(Tag/refs for 2nd RIG)* | |

The file depicted in this figure contains the same raster image information as the file in Figure 2e, "Physical Representation of Data Objects," but the information is organized into two sets. Note that there is only one palette (DFTAG\_IP8/1) and that it is included in both groups.

### General Features of Groups

Figure 5a, "Physical Organization of Sample RIG Groupings," also illustrates a number of important general features of groups:

* The contents of a group must be consistent with one another. Since the palette (DFTAG\_IP8) is designed for use with 8-bit images, the image must be an 8-bit image.
* An application program can easily process all of the images in the file by accessing the groups in the file. The non-RIG information in the example can be used or ignored, depending on the needs and capabilities of the application program.
* There is usually more than one way to group sets. For example, an extra copy of the image palette (DFTAG\_IP8) could have been stored in the file so that each grouping would have its own image palette. That is not necessary in this instance because the same palette is to be used with both images. On the other hand, there are two image dimension records in this example, even though one would suffice.
* Group status does not alter the fundamental role of an HDF object; it is still accessible as an individual data object despite the fact that it also belongs to a larger set.
* A group provides an index of the members of a set. There is nothing to prevent the imposition of other groupings (indexes) that provide a different view of the same collection of data objects. In fact, HDF is designed to encourage the addition of alternate views.

The following sections formally describe raster image sets (RIS), scientific data sets (SDS), Vsets, and several related groups. The last section of this chapter discusses an obsolete structure known as the raster-8 set.

## Raster Image Sets (RIS)

The raster image set (RIS) provides a framework for storing images and any number of optional image descriptors. An RIS always contains a description of the image data layout and the image data. It may also contain color look-up tables, aspect ratio information, color correction information, associated matte or other overlay information, and any other data related to the display of the image.

### Raster Image Groups (RIG)

Tying everything together is the raster image group (RIG, see Figure 5a, "Physical Organization of Sample RIG Groupings," and the related discussion for an example). An RIG contains a list of tag/refs that point in turn to the data objects that make up and describe the image.

The number of entries in an RIG is variable and most of the descriptive information is optional. Complex applications may include references to image-modifying data, such as the color table and aspect ratio, along with the reference to the image data itself. Simple applications may use simple application-level calls and ignore specialized video production or film color correction parameters.

THG currently supports two RIG calling interfaces: RIS8 and RIS24. These interfaces are described in the document NCSA HDF Calling Interfaces and Utilities for Versions 3.2 and earlier and in the HDF User's Guide and HDF Reference Manual for Versions 3.3 and 4.x.

### RIS Tags

RIS implementations must fully support all of the tags presented in Table 5a.

* RIS Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_RIG | Raster image group |
| DFTAG\_ID | Image dimension record |
| DFTAG\_RI | Raster image data |

With these tags, images can be stored and read from HDF files at any bit depth, with several different component ordering schemes. As illustrated in Figure 5b, the RIG tag points to the collection of tag/refs that fully describe the RIS. The data element attached to the tag DFTAG\_ID specifies the dimensions of the image, the number type of the elements that make up its pixels, the number of elements per pixel, the interlace scheme used, and the compression scheme used, if any. The data element attached to the tag DFTAG\_RI contains the actual raster image data.

* RIS Tags

The tags listed in Table 5C identify optional RIS information such as color properties and aspect ratio. Note that the RI interface supports only DFTAG\_LUT at this time; the other tags in Table 5C are defined but the interfaces have not been implemented.



* Optional RIS Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_XYP | XY position of image |
| DFTAG\_LD | Look-up table dimension record |
| DFTAG\_LUT | Color look-up table for non true-color images |
| DFTAG\_MD | Matte channel dimension record |
| DFTAG\_MA | Matte channel data |
| DFTAG\_CCN | Color correction factors |
| DFTAG\_CFM | Color format designation |
| DFTAG\_AR | Aspect ratio |
| DFTAG\_MTO | Machine-type override |

Figure 5c illustrates the structure of an RIS that contains an image palette (DFTAG\_IP8).

* RIS Tags for Sets Containing a Palette

### Raster Image Compression



HDF currently supports the following raster image compression tags:

DFTAG\_RLE Run-length encoding

DFTAG\_IMCOMP Aerial averaging

DFTAG\_JPEG JPEG compression

RIG support does not require support for all compression tags. Be sure to provide a suitable error message to the user when an unknown compression tag is encountered.

Since new forms of data compression can be added to HDF raster images, incompatibilities can arise between old libraries and files created by newer libraries. For example, HDF Versions 3.3 and later include JPEG compression for images. A JPEG-compressed raster image in a file created by an HDF Version 4.1 library cannot be read by an HDF Version 3.2 library.

## Scientific Data Sets

The scientific data set (SDS) provides a framework for storing multidimensional arrays of data with descriptive information that enhances the data. Current specifications support the following types of numbers in SDS arrays.

* 8-bit, 16-bit, and 32-bit signed and unsigned integers
* 32-bit and 64-bit floating point numbers

Data in an SDS can be stored either as two's complement big endian integers, as IEEE Standard floating point numbers, or in native mode, the format used by the machine from which they were written.

The user interface for storing and retrieving SDSs is fully described in the document NCSA HDF Calling Interfaces and Utilities for Versions 3.2 and earlier and in the HDF User's Guide and HDF Reference Manual for Versions 3.3 and 4.x.

### Backward and Forward Compatibility

One concerns in HDF development is always to maximize backward and forward compatibility; as much as possible, any application written to use HDF should be able to read data files written with an older or a newer version of the libraries. To maximize this compatibility, the HDF developement team had to consider the following factors in upgrading the SDS capabilities:

* Support for future variations (e.g., new number types, data compression, and new physical arrangements for SDS storage)
* Older versions of the library should be able to read new data files if the data itself can be interpreted by the older version. To do so, the older version must be able to determine whether the data in a given data object will be comprehensible to it. For example, if a newly created file contains 32-bit IEEE floating point or Cray floating point data objects, older versions of the library should be able determine that fact then read and interpret the data.
* New libraries must be able to read and interpret files created by older versions.

Unfortunately, such compatibility concerns yield an SDS structure somewhat more complex than would otherwise be the case. Two examples illustrate the problem:

* HDF 3.2 development had to accommodate the fact that HDF Version 3.1 and previous versions only supported 32-bit IEEE floating-point numbers and Cray floating point numbers in SDSs. SDSs in HDF versions since Version 3.2 support 8-bit, 16-bit, and 32-bit signed and unsigned integers, 32-bit and 64-bit floating-point numbers, and the local machine format (native mode) for all supported architectures.
* HDF 3.3 includes support for the netCDF data model, which involved the creation of an entire new structure for supporting netCDF objects, based on Vgroups and Vdatas. At the same time, a goal of HDF 3.3 was to harmonize the SDS and the netCDF data model, which was best accomplished by storing SDS objects in the same way that netCDF objects are stored. In order to maintain backward compatibility, two structures had to be created for every SDS or netCDF object: one that could be recognized by older HDF libraries, and the new structure.

In the following sections we describe how the first problem was solved. A later issue of this manual will describe how the second problem was addressed.

### Internal Structures

The SDS capability was substantially enhanced for HDF Version 3.2. Previous versions employed a structure known as a scientific data group (SDG); Version 3.2 and subsequent versions use the numeric data group (NDG). To accommodate the enhanced structure and to remain compatible with previous releases, the current HDF library supports the following scientific and numerical data groups:

SDGs Created by old libraries and containing 32-bit IEEE and Cray floating-point data.

NDGs Created by the newer libraries (Version 3.2 and later) and containing any acceptable floating-point or non-floating-point data. This data group will not be recognized by old libraries.

The NDG structure supports 8-bit, 16-bit, and 32-bit signed and unsigned integers, and 32-bit and 64-bit floating-point numbers. It also supports native mode, data sets written to HDF files in the local machine format.

SDG-like NDGs

Created by the new library and containing IEEE 32-bit floating-point data only. The old libraries will recognize and interpret these numerical data groups correctly.

The following sections describe the SDG, NDG, and SDG-like NDG structures.

### SDG Structures

SDGs must contain at least the data objects listed in Table 5D.

* Required SDG Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_SDG | Scientific data group. |
| DFTAG\_SDD | Dimension record for array-stored data. Includes the rank (number of dimensions), the size of each dimension, and the tag/refs representing the number type of the array data and of each dimension.  All SDG number types are 32-bit IEEE floating-point. |
| DFTAG\_SD | Scientific data. |

In addition to the required data objects listed above, SDGs may contain any of the objects listed in Table 5E. Note that the optional data objects are the same for SDGs, NDGs, and SDG-like NDGs; the only differences are the number types that may be used.

* Optional SDG, NDG, and SDG-like NDG Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_SDS | Scales of the different dimensions. To be used when interpreting or displaying the data (32-bit floating point numbers only for SDGs and SDG-like NDGs). |
| DFTAG\_SDL | Labels for all dimensions and for the data. Each of the dimension labels can be interpreted as an independent variable; the data label is the dependent variable. |
| DFTAG\_SDU | Units for all dimensions and for the data. |
| DFTAG\_SDF | Format specifications to be used when displaying values of the data. |
| DFTAG\_SDM | Maximum and minimum values of the data. (32-bit floating point numbers only for SDGs and SDG-like NDGs.) |
| DFTAG\_SDC | Coordinate system to be used when interpreting or displaying the data. |

As illustrated in Figure 5d, the SDG tag points to the collection of tag/refs that define the SDG.

* SDG Structure

### NDG Structures



NDGs must contain at least the data objects listed in Table 5F

* Required NDG Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_NDG | Numerical data group. |
| DFTAG\_SDD | Dimension record for array-stored data. Includes the rank (number of dimensions), the size of each dimension, and the tag/refs representing the number types of the data and of each dimension.  In HDF 3.2 , the number types of dimension scales must be the same as that of the array-stored data. Later implementations allow dimension scales to be typed separately. |
| DFTAG\_SD | Scientific data. |
| DFTAG\_NT | Number type of the data set. Default is the most recent DFSDsetNT() setting. If DFSDsetNT() has not been called, the default will be 32-bit IEEE floating-point. |

In addition to these required data objects, an NDG may contain any of the data objects listed in Table 5E, "Optional SDG, NDG, and SDG-like NDG Tags," on page 46.

As illustrated in Figure 5e, the basic NDG and SDG structures are identical. The first clue to the difference is that the NDG tag replaces the SDG tag. This is a flag to prevent older libraries from stumbling over the more important difference; the NDG data element can accommodate data that pre-Version 3.2 libraries cannot interpret. The new tag ensures that older libraries will not recognize the data object and thus will not try to interpret the new data types. For example, NDG data can include number types or a data compression scheme that a pre-Version 3.2 library will not recognize.

* NDG Structure

### SDG-like NDG Structures



As we have said earlier,

* SDGs, the SDS grouping structure available prior to HDF Version 3.2, could include only 32-bit floating point and Cray floating point numbers.
* NDGs, available since Version 3.2, can include 8-bit, 16-bit, and 32-bit signed and unsigned integers, and 32-bit and 64-bit floating point numbers.
* SDG-like NDGs, also available since Version 3.2, distinguish SDSs that can still be read by the older versions of the library.

This backward compatibility is achieved by examining every SDS that is written to an HDF file. If the SDS is compatible with older libraries, it is written to the file with both SDG and NDG structures. If it is not compatible with older libraries, only the NDG structure is used.

Table 5G lists the objects that SDG-like NDGs must contain.

* Required SDG-like NDG Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_NDG | Numerical data group. |
| DFTAG\_SDG | Scientific data group. |
| DFTAG\_SDLNK | The NDG and SDG linked to the scientific data set in this group. |
| DFTAG\_SDD | Dimension record for array-stored data. Includes the rank (number of dimensions), the size of each dimension, and the tag/refs representing the number types of the data and of each dimension.  In an SDG-like NDG, the number types are all 32-bit IEEE floating-point. |
| DFTAG\_SD | Scientific data. |

SDG-like NDGs can include the same optional data objects as described for SDGs and NDGs in Table 5E, "Optional SDG, NDG, and SDG-like NDG Tags," on page 46.

Figure 5f illustrates the SDG-like NDG structure.

* SDG-like NDG Structure

### Compatibility with Future NDG Structures



Future HDF releases will probably support additional optional SDS features. These features will fall into the following categories:

Optional and compatible features

Optional features that are compatible with older HDF versions even though they may not be supported in the older libraries.

For example, a new time stamp attribute might be added. The time stamp would not be understood by older libraries, but it would not render them unable to read the SDS data either

Optional and incompatible features

Optional new features that may render the data unreadable by older HDF libraries.

For example, a compression attribute could be added. Older HDF libraries that contain no compression routines would not be able to read the compressed data.

A tag numbering convention has been developed to address this problem:

Required tags

These tags are listed in Table 5D, "Required SDG Tags," on page 46; Table 5F, "Required NDG Tags," on page 47; and Table 5G, "Required SDG-like NDG Tags," on page 48. All SDSs must contain all of the tags in at least one of these sets. (See Chapter , "Tag Specifications," for the assigned tag numbers.)

Optional-incompatible tags

Tags for new SDS features that might render the data set unreadable by older libraries are each assigned a number t that falls in a special range determined by the constants DFTAG\_EREQ and DFTAG\_BREQ. That is, t must have a value such that DFTAG\_EREQ < t < DFTAG\_BREQ. When old software encounters a tag in this range that it is not able to interpret, it should not process the group.

Optional-compatible tags

These tags can have any valid tag number not allocated to one of the other two categories.

## Vsets, Vdatas, and Vgroups

Vsets, Vdatas, and Vgroups enable users to create their own grouping structures. Unlike RIGs, SDGs, and NDGs, HDF imposes no required structure; they are implemented almost entirely at the user level and are not specified in detail in HDF or in this document.[[1]](#footnote-1) The only specifications define DFTAG\_VG, DFTAG\_VH, and DFTAG\_VS and the formats of their respective data elements. A detailed discussion similar to that for the other grouping structures is, therefore, inappropriate here. Detailed information regarding the DFTAG\_VG, DFTAG\_VH, and DFTAG\_VS tags can be found in Chapter , "Tag Specifications." Conceptual and usage information can be found in the document NCSA HDF Vset Version 2.0 for HDF Versions 3.2 and earlier and in the HDF User's Guide and the HDF Reference Manual for HDF Versions 3.3 and 4.x.

* Illustration of a Vset

An HDF Vset can contain any logical grouping of HDF data objects within an HDF file. Vsets resemble the UNIX file system in that they impose a basically hierarchical structure but also allow cross-linked data objects. Unlike SDSs and RISs, Vsets have no prespecified content or structure; users can use them to create structural relationships among HDF objects according to their needs. Figure 5g illustrates a Vset.



A Vset is identified by a Vgroup, an HDF object that contains information about the members of the Vset. The tag DFTAG\_VG identifies the Vgroup which contains the tag/refs of its members, an optional user-specified name, an optional user-specified class, and fields that enable the Vgroup to be extended to contain more information.

The only required Vgroup tag is the tag that defines the Vgroup itself.

* The Vgroup Tag

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_VG | Vgroup |

Vgroups are fully described in the document NCSA HDF Vset, Version 2.0 for Versions 3.2 and earlier and in the HDF User's Guide and HDF Reference Manual for Versions 3.3 and 4.x.

## The Raster-8 Set (Obsolete)

Current HDF versions use the raster image set (RIS) to manage raster images. But before the RIS was implemented, a simpler, less flexible set called the raster-8 set was used for storing 8-bit raster images. This set is no longer supported in the HDF software, although it may turn up in some older HDF files.[[2]](#footnote-2)

### Raster-8 Sets

The raster-8 set is defined by a set of tags that provide the basic information necessary to store 8-bit raster images and display them accurately without requiring the user to supply dimensions or color information. The raster-8 set tags are listed in Table 5I.

* Raster-8 Set Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_RI8 | 8-bit raster image data |
| DFTAG\_CI8 | 8-bit raster image data compressed with run-length encoding |
| DFTAG\_II8 | IMCOMP compressed image data |
| DFTAG\_ID8 | Image dimension record |
| DFTAG\_IP8 | Image palette data |

Software that does not support DFTAG\_CI8 or DFTAG\_II8 must provide appropriate error indicators to higher layers that might expect to find these tags.

### Compatibility Between Raster-8 and Raster Image Sets

To maintain backward compatibility with raster-8 sets, the RIS interface stores tag/refs for both types of sets. For example, if an image is stored as part of a raster image set, there is one copy each of the image dimension data, the image data, and the palette data. But there were two sets of tag/refs pointing to each data element: one for the RIS and one for the raster-8 set. The image data, for example, is associated with the tags DFTAG\_RI8 and DFTAG\_RI.

|  |
| --- |
| Note: Raster-8 set support will not be maintained in future HDF releases. |

Note that future HDF releases will phase out support for the raster-8 set. Therefore, new software should not expect to find both raster-8 and RIS structures supporting 8-bit raster images. Eventually, only RIS structures will be supported.

## Deleted information from "Vsets, Vdatas, and Vgroups:"

A table structure known as a Vdata is often used as a data object in connection with Vsets. The data in a Vdata is organized into fields. Each field is identified by a unique fieldname. The type of each field may be any of the data types supported by the SDS interface: 8-, 16-, and 32-bit integers (signed or unsigned), and 32- and 64-bit floating point numbers. Several fields of different types may exist within a Vdata.

The use of Vdatas requires two tags, DFTAG\_VS and DFTAG\_VH, listed in Table 5J. The flexibility of the Vgroup structure allows the use of any HDF tag.

* Optional Vgroup Tags

|  |  |
| --- | --- |
| Tag | Contents of Data Element |
| DFTAG\_VS | Vdata. |
| DFTAG\_VH | Vdata description. |
| Any HDF tag | The flexibility of the Vgroup structure allows the optional use of any HDF tag. |

1. Specialists in various fields are developing application program interfaces (APIs) that are becoming accepted standard interfaces within their fields. Since these APIs are implemented with high level HDF functionality and using the standard HDF user interface, they are user-level applications from the HDF development team's point of view. From the final enduser's point of view, however, these APIs create a new level of user interface. When necessary, technical specifications for these APIs and the associated interfaces will be presented by the specialized developers. [↑](#footnote-ref-1)
2. In fact, during the first three years that RIS was used, the HDF software stored raster images in both RIS and raster-8 sets. [↑](#footnote-ref-2)