Appendices

* Reserved HDF Tags
* Overview

This appendix includes tables containing brief descriptions of most of the tags that have been reserved for general use. This list will be expanded in future editions to include new tags as they are assigned. A more detailed description of the tags can be found in the *HDF Specification and Developer’s Guide*. Also see the *HDF Specification and Developer’s Guide* for a description of extended tags, which are not discussed in this Appendix.

Each table contains a list of tags within one category. The titles of the tables, with a functional description of each table, are:

* **Table A: The HDF Utility Tags.** Used by the HDF utilties.
* **Table B: The HDF General Raster Image Tags.** Used to describe aspects of raster image data.
* **Table C: The HDF Composite Image Tags.** Used to describe aspects of composite image data.
* **Table D: The HDF Scientific Data Set Tags:** Used to describe aspects of scientific data set (SDS) data.
* **Table E: The HDF Vset Tags.** Used to describe aspects of HDF Vset data.
* **Table F: The Obsolete HDF Tags:** Used to describe aspects of HDF data elements that have been replaced by newer tags or discontinued.
* Tag Types and Descriptions

The following tables have five columns:

Tag Name contains the abbreviated symbolic names of tags that are often used in an augmented form in HDF programs.

Short Description contains a brief (four word maximum) description of the tag that is commonly used to describe to the tag in HDF manuals and in-line code documentation.

Data Size describes the type of data that is associated with the tag and, where possible, lists the data size.

Tag Value lists the numeric value of the tag symbol in the hdf.h header file.

Long Description contains a general description of the tag.

In the tables, the term String refers to a sequence of ASCII characters with the null byte possibly occurring at the end, but nowhere else. The term Text also refers to a sequence of ASCII characters, but it may contain null characters anywhere in the sequence. An n in the Data Size column describes a data unit of variable-length. For more detailed descriptions of these units of data, refer to the appropriate tag entry in the *HDF Specification and Developer’s Guide*.

* The HDF Utility Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_NULL | No Data | None | 001 | Used for place holding and filling up empty portions of the Data Descriptor Block. |
| DFTAG\_VERSION | Library Version Number | 4 bytes + string | 030 | Specifies the latest version of the HDF library used to write to the file. |
| DFTAG\_NT | Number Type | 4 bytes | 106 | Used by any other element in the file to specifically indicate what a numeric value looks like. |
| DFTAG\_MT | Machine Type | 0 bytes | 107 | Specifies that all unconstrained or partially constrained values in this HDF file are of the default type for that hardware. |
| DFTAG\_FID | File Identifier | String | 100 | Points to a string that the user wants to associate with this file. This supports the inclusion of a user-supplied title for the file. |
| DFTAG\_FD | File Descriptor | Text | 101 | Points to a block of text describing the overall file contents. It is intended to be user-supplied comments about the file. |
| DFTAG\_TID | Tag Identifier | String | 102 | Provides a way to determine the meaning of a tag stored in the file. |
| DFTAG\_TD | Tag Descriptor | Text | 103 | Similar to DFTAG\_TD, but allows more text to be included. |
| DFTAG\_DIL | Data Identifier Label | String | 104 | Associates the string with the Data Identifier as a label for whatever the identifier points to. By including DILs, any data element can be given a label for future reference. For example, this tag is often used to give titles to raster image data sets. |
| DFTAG\_DIA | Data Identifier Annotation | Text | 105 | Associates the text block with the Data Identifier as an annotation for whatever that Data Identifier points to. With DIAs, and Data Identifier can have a lengthy, user-provided description of why that particular data element is in the file. |
| DFTAG\_RLE | Run-length Encoding | 0 bytes | 011 | Specifies that run-length encoding (RLE) is used to compress a raster image. |
| DFTAG\_IMC | IMCOMP  Compression | 0 bytes | 012 | Specifies that IMCOMP compression is used to compress a raster image. |
| DFTAG\_JPEG | 24-bit JPEG  Compression | n bytes | 013 | Provides header information for 24-bit JPEG-compressed raster images. |
| DFTAG\_GREYPEG | 8-bit JPEG  Compression | n bytes | 014 | Provides header information for 8-bit JPEG-compressed raster images. |

* The HDF General Raster Image Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_RIG | Raster Image Group | n\*4 bytes | 306 | Lists the Data Identifiers (tag/reference number pairs) that uniquely describe a raster image set. |
| DFTAG\_ID | Image Dimension | 20 bytes | 300 | Defines the dimensions of the two-dimensional array the corresponding RI tag refers to. |
| DFTAG\_LD | LUT Dimension | 20 bytes | 307 | Defines the dimensions of the two-dimensional array the corresponding LUT tag refers to. |
| DFTAG\_MD | Matte Dimension | 20 bytes | 308 | Defines the dimensions of the two-dimensional array the corresponding MA tag refers to. |
| DFTAG\_RI | Raster Image | x\*y bytes | 302 | Points to a raster image data set. |
| DFTAG\_CI | Compressed Image | n bytes | 303 | Points to a compressed raster image data set. |
| DFTAG\_LUT | Lookup Table | n bytes | 301 | Table to be used by the hardware for the purpose of assigning RGB or HSV colors to data values. |
| DFTAG\_MA | Matte Data | n bytes | 309 | Points to matte data. |
| DFTAG\_CCN | Color Correction | n bytes | 310 | Specifies the gamma correction for the raster image and color primaries used in the generation of the image. |
| DFTAG\_CFM | Color Format | String | 311 | Indicates the interpretation to be given to each element of each pixel in a raster image. |
| DFTAG\_AR | Aspect Ratio | 4 bytes | 312 | Indicates the aspect ratio of the image. |
| DFTAG\_XYP | XY Position | 8 bytes | 500 | Specifies the screen X-Y coordinate for raster image sets. (Also used for composite image sets - See the entry for DFTAG\_XYP in Table 12.6) |

* The HDF Composite Image Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_DRAW | Draw | n\*4 bytes | 400 | Specifies a list of Data Identifiers (tag/reference number pairs) which define a composite image. |
| DFTAG\_XYP | XY Position | 8 bytes | 500 | Specifies the screen X-Y coordinate for composite image sets. (Also used for raster image sets - See the entry for DFTAG\_XYP in Table 12.5) |
| DFTAG\_RUN | Run | n bytes | 401 | Identifies code that is to be executes as a program or script. |
| DFTAG\_T14 | Tektronix 4014 | n bytes | 602 | Used as a vector image tag**.** Points to a Tektronix 4014 data. The bytes in the data field, when read and sent to a Tektronix 4014 terminal, will be displayed as a vector image. |
| DFTAG\_T10S | Tektronix 4015 | n bytes | 603 | Used as a vector image tag.Points to a Tektronix 4015 data. The bytes in the data field, when read and sent to a Tektronix 4015 terminal, will be displayed as a vector image. |

* The HDF Scientific Data Set Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_NDG | Numeric Data Group | n\*4 bytes | 720 | Lists the Data Identifiers (tag/reference number pairs) that describe a scientific data set. Supersedes DFTAG\_SDG. |
| DFTAG\_SDD | SDS Dimension Record | n bytes | 701 | Defines the rank and dimensions of the array the corresponding SD refers to. |
| DFTAG\_SD | Scientific Data | Real Number | 702 | Points to scientific data. |
| DFTAG\_SDS | SCales | Real Number | 703 | Identifies the scales to be used when interpreting and displaying data. |
| DFTAG\_SDL | Labels | String | 704 | Labels all dimensions and data. |
| DFTAG\_SDU | Units | String | 705 | Displays units for all dimensions and data. |
| DFTAG\_SDF | Formats | String | 706 | Displays formats for axes and data. |
| DFTAG\_SDM | Maximum/minimum | 2 Real Numbers | 707 | Displays the maximum and minimum values for the data. |
| DFTAG\_SDC | Coordinate system | String | 708 | Displays the coordinate system to be used in interpreting data. |
| DFTAG\_SDLNK | SDS Link | 8 bytes | 710 | Links and old-style DFTAG\_SDG and a DFTAG\_NDG in cases where the DFTAG\_NDG meets all criteria for a DFTAG\_SDG. |
| DFTAG\_CAL | Calibration Information | 36 bytes | 731 | The calibration record for the corresponding DFTAG.SD. |
| DFTAG\_FV | Fill Value | n bytes | 732 | The value which has been used to indicate unset values in the corresponding DFTAG\_SD. |

* The HDF Vset Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_VG | Vgroup | 14+n bytes | 1965 | Provides a general-purpose grouping structure. |
| DFTAG\_VH | Vdata Description | 22+n bytes | 1962 | Provides information necessary to process a DFTAG\_VS. |
| DFTAG\_VS | Vdata | n bytes | 1963 | Contains a block of data that is to be interpreted according to the information in the corresponding DFTAG\_VH. |

* The Obsolete HDF Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag Name | Short Description | Data Size | Tag Value | Long Description |
| DFTAG\_IDS | Image Dimension-8 | 4 bytes | 200 | Two 16-bit integers that represent the width and height of an 8-bit raster image in bytes. |
| DFTAG\_IP8 | Image Palette-8 | 768 bytes | 201 | A 256 x 3 byte array representing the red, green and blue elements of the 256-color palette respectively. |
| DFTAG\_RI8 | Raster Image-8 | x\*y bytes | 202 | A row-oriented representation of the elementary 8-bit image data. |
| DFTAG\_CI8 | Compressed Image-8 | n bytes | 203 | A row-oriented representation of the elementary 8-bit raster image data, with each row compressed using a form of run-length encoding. |
| DFTAG\_II8 | IMCOMP Image-8 | n bytes | 204 | A 4:1 8-bit raster image, compressed using the IMCOMP algorithm. |
| DFTAG\_SDG | Scientific Data Group | n\*4 bytes | 700 | List the Data Identifiers (tag/reference number pairs) that uniquely describe a scientific data set. |
| DFTAG\_SDT | Transpose | 0 bytes | 709 | Indicates that data is transposed in the file. |

* HDF Installation Overview
* General HDF Installation Overview
* Acquiring the HDF Library Source

You may obtain the HDF source code and/or selected binaries at no charge from The HDF Group's server:

http://www.hdfgroup.org/products/hdf4

http://www.hdfgroup.org/release4/obtain.html

For reference, the unpacked HDF source code can be found at

ftp://ftp.hdfgroup.org/HDF/HDF\_Current/src/unpacked/.

* Building the HDF Library Source

For instructions on building HDF from the source code, please refer to the INSTALL file in the top directory of the unpacked HDF source tree.

* Attributes in HDF

This Appendix gives an overview of attributes in HDF and describes some issues in the library regarding attributes. The information is more helpful when users are working with files produced by older versions of the library.

* Attribute Overview

Attributes are optional components in the HDF data model. They can be used to describe the nature and/or the intended usage of various HDF elements. This type of information is sometimes called user-created metadata because it is data about data. The HDF elements that can be assigned with attributes include:

* File, data set, and dimension in SD API
* File and raster image in GR API
* Vgroup in V API
* Vdata and vdata field in VS API

At the creation, an HDF attribute requires a name, data values, number type, and number of values. The attribute name is an ASCII string of any length from 1 to H4\_MAX\_NC\_NAME (or 256). The attribute data contains one or more values, in which case all the values must have the same number type as defined at the time the attribute is created. Attributes take the form label=value, where label is the attribute’s name and value is the attribute’s data. Number of values declares how many data entries the attribute has. The number type can be any type supported by the HDF library. These number types are listed in Table 1A, "Number Type Definitions" in Section I of the HDF4 Reference Manual.

For each attribute, an attribute count is maintained that identifies the number of values in the attribute. Each attribute has a unique attribute index, the value of which ranges from 0 to the total number of attributes minus 1. The attribute index is used to locate an attribute in the object which the attribute is attached to. Once the attribute is identified, its values and information can be retrieved.

There are two types of attributes in HDF: predefined ***attributes*** and user-defined attributes.

Predefined attributes have reserved names and, in some cases, predefined number types and/or number of data entries. Predefined attributes are useful because they establish conventions that applications can depend on. They were first introduced in DFSD interface and later in the SD interface. They are further described in Section 3.10, "Predefined Attributes," of the HDF User’s Guide. The GR interface was added in 1995 and has only one predefined attribute: FILL\_ATTR, which is described in Section 8.10.1, "Predefined GR Attributes," of the HDF User’s Guide.

User-defined attributes are defined by the calling program and contain auxiliary information about the element to which the attributes attach. HDF library provides in each interface of SD, GR, V, and VS a set of functions to add and access attributes. They are fully described in the associated chapters.

* Underlaying storage issues

In general, users should not need the details described in this section, unless one is working with older HDF files (circa prior to 1993) and with raw data which relies on the knowledge of data layout in the file. The inclusion of this section in this User’s Guide was prompted by the HDF4 File Content Map Project because various API functions being added to support this project require explanation that involves the layout of attributes in the file.

In the early years of HDF, in addition to the predefined attributes such as label, unit, and format, annotations were used to attach metadata to an HDF element such as data set and raster image. When the library was expanded to include user-defined attributes to SD and GR interfaces, metadata once stored as an annotation could be more conveniently stored as an attribute. This expansion introduced the difference in the ways predefined attributes were stored in DFSD interface and in SD/GR interfaces. The user-defined attribute feature then extended into the V and VS interfaces. Along the way, an incompatibility was inadvertently produced in the storage of attributes and their information. The next sections briefly explains these issues and their effects.

* Predefined Attributes in DFSD API

Beginning in 1993, when the SD interface and user-created attribute were introduced, an attribute has been stored in a vdata of class \_HDF\_ATTRIBUTE (or "Attr0.0",) regardless it is a predefined or user-created attribute. However, prior to this period, there were only predefined attributes in DFSD API and they can be assigned to a data set or a dimension. This early predefined attribute of the data set is stored using tag/ref approach, that is, a pair of tag and ref would point to a string containing the values of the data set’s attribute and the dimensions’ attributes. The dimension attributes are stored following the SDS attribute. All attributes are separated by null characters. For example, in file myfile, there is a two-dimensional data set. The SDS and its dimensions were assigned with pre-defined attributes as followed:

Data set: label = "SDS label", unit = "SDS unit", format = <no attribute assigned>

Dimension 1: label = "Dim1 label", unit = <no attribute assigned>, format = "Dim1 format"

Dimension 2: label = "Dim2 label", unit = "Dim2 unit", format = "Dim2 format"

In the file, the attributes’ values are stored as followed:

Data set’s label attribute tag/ref (DFTAG\_SDL/<ref#>)

| (point to)

--> "SDS label<null>Dim1 label<null>Dim2 label<null>"

Data set’s unit attribute tag/ref (DFTAG\_SDU/<ref#>)

| (point to)

--> "SDS unit<null><null>Dim2 unit"

Data set’s format attribute tag/ref (DFTAG\_SDF/<ref#>)

| (point to)

--> "<null>Dim1 format<null>Dim2 format"

A complete list of pre-defined attribute tags are provided in Table AG below.

* Pre-defined Attributes in the DFSD and SD APIs

|  |  |  |  |
| --- | --- | --- | --- |
| Tag Name | Description | Data Size | Applicable to |
| DFTAG\_SDL | Labels | String | SDS and dimensions |
| DFTAG\_SDU | Units | String | SDS and dimensions |
| DFTAG\_SDF | Formats | String | SDS and dimensions |
| DFTAG\_SDM | Maximum/minimum | 2 Real Numbers | Only SDS |
| DFTAG\_SDC | Coordinate system | String | Only SDS |
| DFTAG\_CAL | Calibration Information | 36 bytes | Only SDS |
| DFTAG\_FV | Fill Value | n bytes | Only SDS |

The HDF library handles the situation properly, so the difference in storage approaches does not effect general applications, which simply read the values of these predefined attributes. It would only become significant when an application needs to get access to the raw data. The HDF4 File Content Map Project is an example. The raw data of this type of attribute is not accessible by the function SDgetattdatainfo, which was added to support the HDF4 File Content Map Project. Thus, when such an attribute is encountered, SDgetattdatainfo will return the error code DFE\_NOVGREP to the caller, which will in turn call SDgetoldattdatainfo to get the data information of that attribute.

* Vgroup Attribute Without Vsetattr

HDF Version 4.0.2, July 19, 1996, and prior did not support attributes in Vgroup and Vdata as for SD and GR interfaces. However, an application could simulate an attribute for a vgroup by creating and writing a vdata of class \_HDF\_ATTRIBUTE, and then adding that vdata to the vgroup via these calls:

vdata\_ref = VHstoredatam(file\_id, ATTR\_FIELD\_NAME, values, size, type,

attr\_name, \_HDF\_ATTRIBUTE, order);

ret\_value = Vaddtagref (vgroup\_id, DFTAG\_VH, vdata2\_ref);

For simplicity, this type of attributes is referred to as old-style attributes in this document.

A vgroup and vdata were having version number as VSET\_VERSION (3). Starting in version 4.1.1, HDF began to support attributes in Vgroup and Vdata interfaces. Applications were able to add and manipulate attributes via public functions such as Vsetattr/VSsetatt, Vgetattr/VSgetattr, Vattrinfo/VSattrinfo,… This type of attributes is referred to as new-style attributes in this document. The version number of a vgroup or a vdata that has new-style attributes got promoted from VSET\_VERSION (3) to VSET\_NEW\_VERSION (4).

In addition, the file format was changed for the vgroup/vdata header to store the number of attributes and the tag/reference number of each attribute. The new attribute API functions use this new information to get access to the attributes, but they are not aware of the old-style attributes. Thus, Vnattrs misses counting them and other functions like Vattrinfo and Vgetattr are unable to get to them.

Starting in version 4.2.6, the library provides the updated functions Vnattrs2, Vattrinfo2, and Vgetattr2 for applications to get access to attributes that were not created by Vsetattr. These functions access both types of attributes. In addition, the HDF library provides the function Vnoldattrs to get the number of old-style attributes in a vgroup. The old-style attributes are likely to present in older files or files that were modified by older applications. Please refer to Section 5.8, "Vgroup Attributes," of the HDF User’s Guide for details on these functions.

* Issue of Missing Palettes

This Appendix describes an issue regarding palettes in old and new raster image interfaces, that is, DF interfaces versus GR interface. The information may be helpful when users are working with files produced by older versions of the library.

* Description

HDF4’s representation of palettes and rasters has evolved over the lifetime of the library. As new representations were adopted, “old-style” representations were also written to the HDF4 file for backward compatibility. This practice occasionally introduce some issues inadvertently. This appendix presents one of those situations.

As discussed in Chapter 2 of the HDF4 Specification and Developer’s Guide, the basic building blocks of an HDF4 file are data objects. A data object has two parts – a data descriptor (DD) and a data element (DE).

The original representation for palettes in HDF4 used DDs with tag DFTAG\_IP8 (201) while the later representation used DDs with tag DFTAG\_LUT (301). Typically, when an HDF4 file has a pair of DDs with [tag 201, ref=R] and [tag 301, ref=R], they are old/new representations of the same palette and both refer to the same DE at the same offset. The DFP APIs and other GR APIs dealing with LUTs expect this behavior.

In some cases, an HDF4 file will have a 201 DD and a 301 DD that have the same reference number, but that refer to DEs at different offsets. In these cases, it is impossible to retrieve all the palette information in the file using the DFP and other GR APIs.

DFPgetpal attempts to read a DD with tag 201 first, and only attempts to read a DD with tag 301 and the same reference number if failure occurs for the first read. This effectively means that DFPgetpal cannot be used to retrieve a DD with tag 301/ref=R if a DD with tag 201/ref=R exists – even if the two DDs reference DEs at different offsets.

Another limitation of the DFP APIs is rooted in the fact that multiple palette DDs (for example 201/ref=2 and 201/ref=3) may refer to the same DE. This can be an issue even if both DDs in any 201/301 pair have the same offset.

DFPnpals returns the number of palette Data Elements in the file, not the number of palette Data Descriptors. Because multiple palette DDs can reference the same DE, the value (N) returned by DFPnpals cannot reliably be used as the upper bound on the number of calls needed to DFPgetpal to retrieve all the palettes in the file. Internally, DFPgetpal gets the next Palette DD and then uses the offset to retrieve the palette data. If multiple Palette DDs reference the same DE, then making N calls to DFPgetpal will not retrieve all of the DDs in the file (there will be more DDs than DEs). If the missed DDs reference DEs that don’t appear in the first N DDs, then the palette data in those DEs will never be read.

* Work-Around

To avoid changing the behavior of existing functions, GRgetpalinfo was added, starting in release 4.2.8, to get access to all palette DDs in an HDF4 file. With this information, the Hgetelement function can be used to retrieve the palette data from the DE associated with each DD. In addition, GRgetpalinfo also provides a way to retrieve palettes that are not associated with any raster image.