* Basic Structure of HDF Files

# Chapter Overview

This chapter introduces and describes the components and organization of Hierarchical Data Format (HDF) files. The components of an HDF file include a file header and a variety of data objects.

## File Header

The first component of an HDF file is the file header (FH), which takes up the first four bytes in an HDF file. The file header is a signature that indicates that the file is an HDF file. Specifically, it is a 4-byte block with the hexadecimal value 0x0E 0x03 0x13 0x01.[[1]](#footnote-1)

To maintain HDF file portability, the characters must be read and written in the exact order shown.

## Data Objects

The basic building block of an HDF file is the data object, which contains both data and information about the data. A data object has two parts: a 12-byte data descriptor (DD) and a data element. Figure 2a illustrates two data objects.

* Two Data Objects



As the names imply, the data descriptor provides information about the data; the data element is the data itself. In other words, all data in an HDF file has information about itself attached to it. In this sense, HDF files are *self-describing* files.

Data Descriptor (DD)

A data descriptor (DD) has four fields: a 16-bit tag, a 16-bit reference number, a 32‑bit data offset, and a 32-bit data length. These are depicted in Figure 2c and are briefly described in Table 2a. Explanations of each part appear in the paragraphs following Table 2a.

* A Data Descriptor (DD)



* Parts of a Data Descriptor

|  |  |  |
| --- | --- | --- |
| Part | Description | |
| Tag/ref | Unique identifier for each data element | |
| (data identifier) | Tag | Type of data in a data element |
|  | Reference number | Number distinguishing data element from others with the same tag |
| Offset | Byte offset of data element from beginning of file | |
| Length | Length of data element in bytes | |

Tag/ref (Data Identifier)

A tag and its associated reference number, abbreviated as tag/ref, uniquely identify a data element in an HDF file. The tag/ref combination is also known as a data identifier.

|  |
| --- |
| **Note:**  Only the full tag/ref uniquely identifies a data element. |

Tag

A tag is the part of a data descriptor that tells what kind of data is contained in the corresponding data element. A tag is actually a 16-bit unsigned integer between 1 and 65535, but every tag is also given a name that programs can refer to instead of the number. If a DD has no corresponding data element, its tag is DFTAG\_NULL, indicating that no data is present. A tag may never be zero.

Tags are assigned by The HDF Group as part of the specification of HDF. The following ranges are to be used to guide tag assignment:

00001 – 32767 Reserved for HDF use

32768 – 64999 User-definable

65000 – 65535 Reserved for expansion of the format

Chapter 9, “Tag Specifications,” provides full specifications for all currently supported HDF tags. Appendix A, “Tags and Extended Tag Labels,” lists the current tag assignments. See Section 3.4, "Some HDF Conventions," for more information on allocating tags.

Reference Number

Tags are not necessarily unique in an HDF file; there may be more than one data element of a given type. Therefore, the data descriptor includes a unique reference number.

Reference numbers are not necessarily assigned consecutively, so you cannot assume that the actual value of a reference number has any meaning beyond providing a means of distinguishing among elements with the same tag. Furthermore, reference numbers are only unique for data elements with the same tag; two 8-bit raster images will never have the same reference number but an 8-bit raster image and a 24-bit raster image might.

Reference numbers are 16-bit unsigned integers.

Data Offset and Length

The data offset states the byte position of the corresponding data element from the beginning of the file. The length states the number of bytes occupied by the data element.

Offset and length are both 32-bit signed integers.

|  |
| --- |
| **Note:**  All offsets are from the beginning of the file; they are not relative. |

This results in a file-size limit of 2 gigabytes.

DD Blocks

Data descriptors are stored physically in a linked list of blocks called data descriptor blocks or DD blocks. The individual components of a DD block are depicted in Figure 2d. All of the DDs in a DD block are assumed to contain significant data unless they have the tag DFTAG\_NULL (no data).

In addition to its DDs, each data descriptor block has a data descriptor header (DDH). The DDH has two fields: a block sizefield and a next blockfield. The block size field is a 16-bit unsigned integer that indicates the number of DDs in the DD block. The next block field is a 32-bit unsigned integer giving the offset of the next DD block, if there is one. The DDH of the last DD block in the list contains a 0 in its next block field.

* Model of a Data Descriptor Block



Since the default number of DDs in a DD block is defined when the HDF library is compiled, changing the default requires recompilation. (The default value, as distributed in the source code and pre-compiled binaries for Version 4.1r4, is 16.)

Data Element

A data element is the raw data portion of a data object. Its data type can be determined by examining its tag, but other interpretive information may be required before it can be processed properly.

Each data element is stored as a set of contiguous bytes starting at the offset and with the length specified in the corresponding DD. (See Figure 2e, "Physical Representation of Data Objects," on page 13.)[[2]](#footnote-2)

Exceptions and Special Cases

Note that there are a few exceptions and special cases to the above standards.

* The data object identified by the tag DFTAG\_MT, for machine type, consists of the tag immediately followed by four number types. Since there can be only one DFTAG\_MT tag in an HDF file and the data can be stored in the DD with the tag, there is no need for a data element. Consequently, the reference number, offset, and length are unnecessary.
* Several tags, specifically DFTAG\_NULL, DFTAG\_JPEG, and DFTAG\_GREYJPEG, serve as binary flags and convey all the required information by the mere fact of their presence in an HDF file. These tags therefore point to no data element and have offset and length values of 0. DFTAG\_NULL indicates a data object containing no data. DFTAG\_JPEG and DFTAG\_GREYJPEG indicate that an associated data object, indicated by a different tag but the same reference number, contains JPEG data image. The descriptions of these tags include a *sink pointer* ( ) in the diagrams in Chapter (which chapter?).



* It is possible to create a tag/ref object then to end access to that object before writing any data or specifying its size. In such cases, the offset and length in the DD block will be set to the invalid offset or invalid length value of 0xFFFFFFFF.

See the related entries in Chapter 9, Tag Specifications, for complete descriptions of these tags.

## Physical Organization of HDF Files

The file header, DD blocks, and data elements appear in the following order in an HDF file:

* File header
* First DD block
* Data elements
* More DD blocks, more data elements, etc., as necessary

These relationships are summarized in Table 2B.

The only rule governing the distribution of DD blocks and data elements within a file is that the first DD block must follow immediately after the file header. After that, the pointers in the DD headers connect the DD blocks in a linked list and the offsets in the individual DDs connect the DDs to the data elements.

* Summary of the Relationships among Parts of an HDF File

|  |  |
| --- | --- |
| Part | Constituents |
| HDF file | FH, DD block, data, DD block, data, DD block, data... |
| FH | 0x0e031301 [32-bit HDF magic number] |
| DD block | DDH, DD, DD, DD, ... |
| DDH | Number of DDs [16 bits], offset to next DD block [32 bits] |
| DD | Tag [16 bits], ref [16 bits], offset [32 bits], length [32 bits] |
| Data | Data element, data element, data element ... |

FH = file header, DD = data descriptor, DDH = DD header

## Sample HDF File

We are now ready to examine a sample file. Consider an HDF file that contains two 400-by-600 8-bit raster images as described in Table 2C.

* Sample Data Objects in an HDF File

|  |  |  |
| --- | --- | --- |
| Tag | Ref | Data |
| DFTAG\_FID | 1 | File identifier: user-assigned title for file |
| DFTAG\_FD | 1 | File descriptor: user-assigned block of text describing overall file contents |
| DFTAG\_LUT | 1 | Image palette (768 bytes) |
| DFTAG\_ID | 1 | *x*- and *y*-dimensions of the 2-dimensional arrays that contain the raster images (4 bytes) |
| DFTAG\_RI | 1 | First 2-dimensional array of raster image pixel data (*x*\**y* bytes) |
| DFTAG\_RI | 2 | Second 2-dimensional array of pixel data (also *x*\**y* bytes) |

Assuming that a DD block contains 10 DDs, the physical organization of the file could be described by Figure 2e.

In this instance, the file contains two raster images. The images have the same dimensions and are to be used with the same palette, so the same data objects for the palette (DFTAG\_IP8) and dimension record (DFTAG\_ID8) can be used with both images.

* Physical Representation of Data Objects

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Item | Offset | Contents |
| Header | FH | 0 | 0e031301 *(HDF magic number, in hexadecimal)* |
| DD block | DDH | 4 | 10 0 |
|  | DD | 10 | DFTAG\_FID 1 130 4 |
|  | DD | 22 | DFTAG\_FD 1 134 41 |
|  | DD | 34 | DFTAG\_LUT 1 175 768 |
|  | DD | 46 | DFTAG\_ID 1 943 4 |
|  | DD | 58 | DFTAG\_RI 1 947 240000 |
|  | DD | 70 | DFTAG\_RI 2 240947 240000 |
|  | DD | 82 | DFTAG\_NULL *(Empty)* |
|  | DD | 94 | DFTAG\_NULL *(Empty)* |
|  | DD | 106 | DFTAG\_NULL *(Empty)* |
|  | DD | 118 | DFTAG\_NULL *(Empty)* |
| Data | Data | 130 | sw3 |
|  | Data | 134 | solar wind simulation: third try. 8/8/88 |
|  | Data | 175 | .... *(Data for the image palette)* |
|  | Data | 943 | 400 600 *(Image dimensions)* |
|  | Data | 947 | .... *(Data for the first raster image)* |
|  | Data | 240947 | .... *(Data for the second raster image)* |

1. 0x0E 0x03 0x13 0x01 is the hexadecimal representation of the characters control-N, control-C, control-S, and control-A, or ^N^C^S^A. [↑](#footnote-ref-1)
2. Some HDF software provides the capability of storing objects as a series of linked blocks or external elements, but this occurs at a higher level. At the lowest level, each object with a tag/ref is stored contiguously. [↑](#footnote-ref-2)