Chapter 2

The HDF Data Models

2.0 Chapter Overview

The HDF library currently supports six different data models, where each data model represents a framework for accessing a different type of data and its associated information. In a sense, each data model can be thought of as a set of tools for customizing the contents an HDF file. Although there is some overlap among tool sets, in most cases each set of tools is limited to operating on data from one data model. As a result of HDF's "tool set modularity," you need only familiarize yourself with the data model specific to your needs.

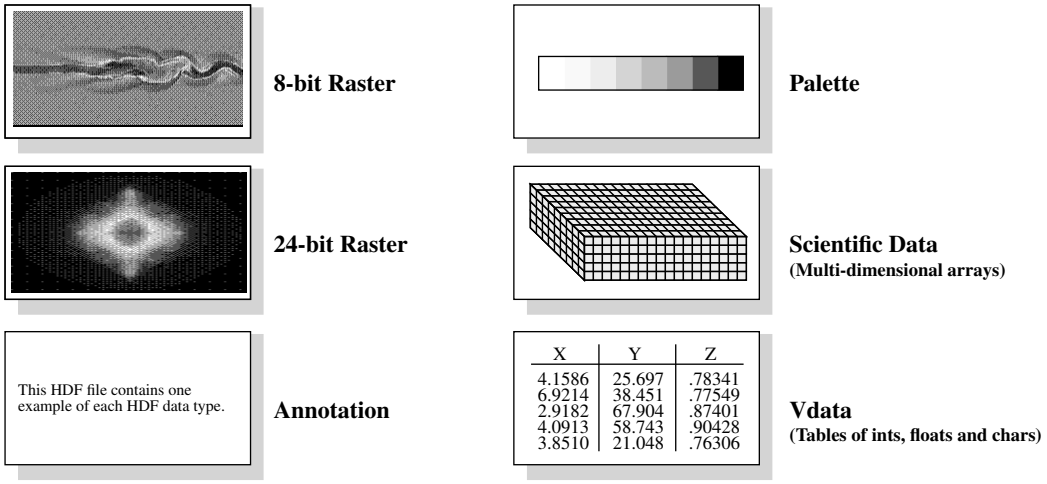
In Chapter 3, you will learn to think of each data model in the following way:

- **The *8-bit raster model*** stores and retrieves 8-bit raster images, their dimensions, and palettes.
- **The *palette model*** stores and retrieves 8-bit palettes outside the 8-bit raster model.
- **The *24-bit raster model*** stores and retrieves 24-bit images and their dimensions.
- **The *scientific data model*** stores and retrieves multi-dimensional arrays of integer or floating-point numbers, their dimensions, number type, and attributes.
- **The *annotation model*** stores and retrieves the text strings used to describe a file or any of the data elements it contains.
- **The *virtual data model*** stores and retrieves multi-variate data stored as records in a table.

2.1 Introducing the Data Models Supported by HDF

The HDF library currently supports six different data models (See FIGURE 2.1a). In practice, a ***data model*** is a framework or data structure used by HDF to store a particular type of data along with its associated information. For example, the Vdata model uses tables to store records containing fields of different number types.

FIGURE 2.1a **The HDF library supports six different data models.**



HDF data models are designed to support only those data elements which are applicable to the group as a whole. In other words, data models are limited to data elements which "make sense" in the context of the group. The 24-bit raster model, for example, will not support palettes or three-dimensional arrays because neither of them are necessary for 24-bit imaging operations.

Recognized groups of data elements are called ***HDF data sets***. All sets have a minimum number of required elements, and a maximum number of optional elements. A raster image set, for example, must contain both a 2-dimensional array of 8-bit numbers and a dimension record, however adding a palette is optional. In the case of 8-bit raster data, HDF limits each set to one 8-bit image array, one dimension record, and one palette. If a raster image set contains all three elements, the set is considered complete.

In the following sections you will learn more about each data model, the type of data it supports, and its acceptable data elements.

2.2 The 8-bit Raster Model

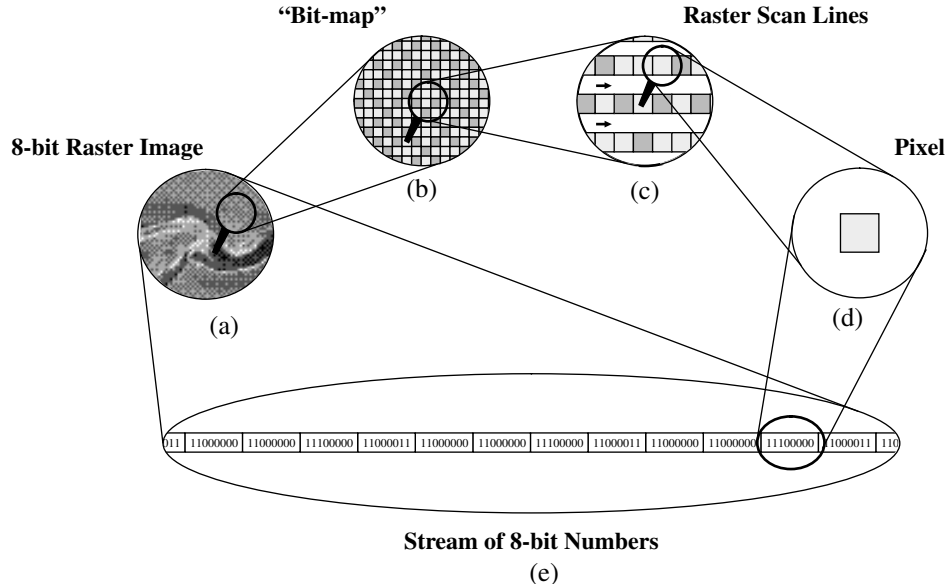
In the days when computers were limited to monochrome displays, determining the value of a pixel required interpreting one bit of information. If the bit was "on" the pixel was white, if the bit was "off" the pixel was black. This simple one-to-one mapping strategy gave rise to the term ***bit-map***. Today the term bit-map is often used to describes any image whose colors or grayscale intensities are stored as n-bits per pixel.

Today, the terms “bit-map” and “raster” are used almost interchangeably. However, the phrase ***raster image*** comes from the application of television technology to bit-map imaging. In a standard raster display, images are scanned out sequentially one line at a time from left to right, top to bottom. An entire raster image is thus formed by scanning a set of horizontal lines each as long as the image is wide.

Although televisions are limited to single image of a standard size, computer monitors are most often used to display multiple images of varying sizes. Consequently, the dimensions for each image must be stored with the image data. Once the dimensions are known however, raster images are easily condensed into a single stream of data. Recreating an image from a single data stream is then simply a matter of painting the appropriate number of pixels per line.

In HDF, an ***8-bit raster image*** is defined as any bit-mapped image whose rows are painted from left to right using pixels whose values are represented by a stream of 8-bit numbers (See FIGURE 2.2a). As a simple example of the mechanics behind 8-bit raster imaging, consider a stream of 8-bit numbers representing raster image data. To display the data as a bit-map, the color associated with the first number in the data stream must appear in the upper left corner of the image. The remainder of the first line is then painted from left to right using as many values from the data stream as necessary to complete the line. The remainder of the rows are similarly painted from left to right, top to bottom until every value in the data stream appears as one pixel in the image.

FIGURE 2.2a **An 8-bit raster image(a) is a bit-map(b) whose rows are painted from left to right(c) using pixels(d) whose values are stored as a single stream of 8-bit numbers(e)**.



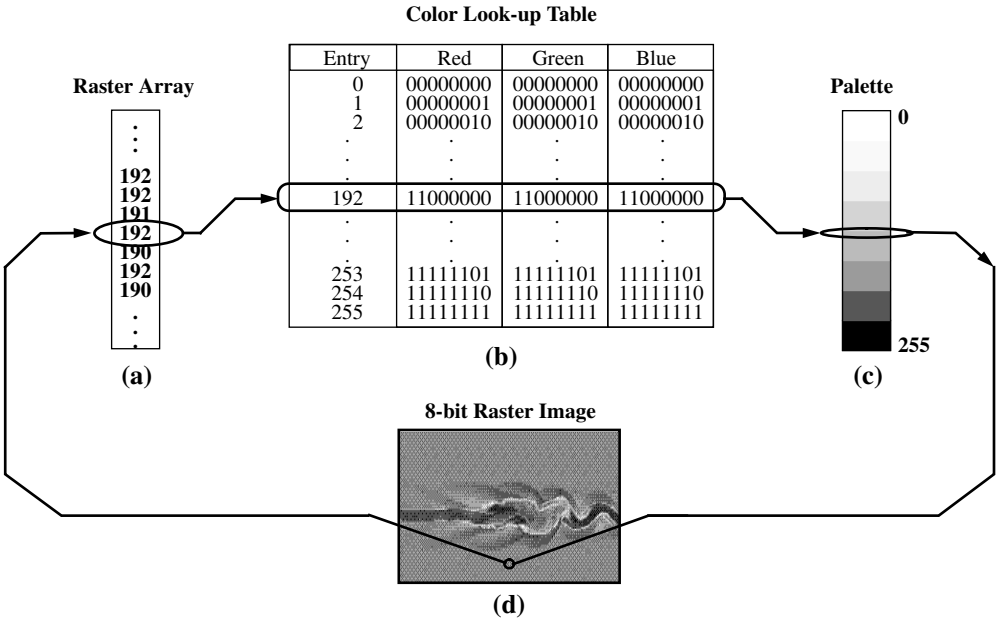
The HDF ***8-bit raster model*** provides a framework for storing 8-bit raster images along with their image dimensions and palettes. Any 8-bit raster image array accompanied by its dimension record is recognized as a ***8-bit raster image set***. A raster image set may also include a palette, however, palettes are optional and may be omitted. To learn more about raster image sets, consult the HDF User’s Guide.

2.3 The Palette Model

Palettes are the means by which color is applied to 8-bit raster image data. Recall from the previous section that an 8-bit raster image is stored as a single stream of data where each pixel value is represented by one 8-bit number. Because pixel values are stored as 8-bit numbers, they are limited to values between 0 and 255.

Every pixel value appearing in an image is assigned a color through a ***color look-up table***. In one sense, pixel values serve as an index for 256 entries stored in a color look-up table (See FIGURE 2.3a). On the other hand, pixel values also represent the color entries in a ***palette***. Whether you prefer to think of assigning colors via the numerical entries in a color look-up table or the color entries in a palette, both represent a means for assigning the same color to every pixel with the same pixel value. For example, every pixel assigned a pixel value of 192, will be painted using the values stored in table entry 192 or the color stored in palette entry 192. If the color entry or its numerical equivalent is changed, every pixel in the image with a value of 192 will also change.

FIGURE 2.3a **For every pixel value in a raster array(a) there is one entry in a color look-up table(b) that will determine its color in the palette(c), and thus its appearance in the resultant image (d).**



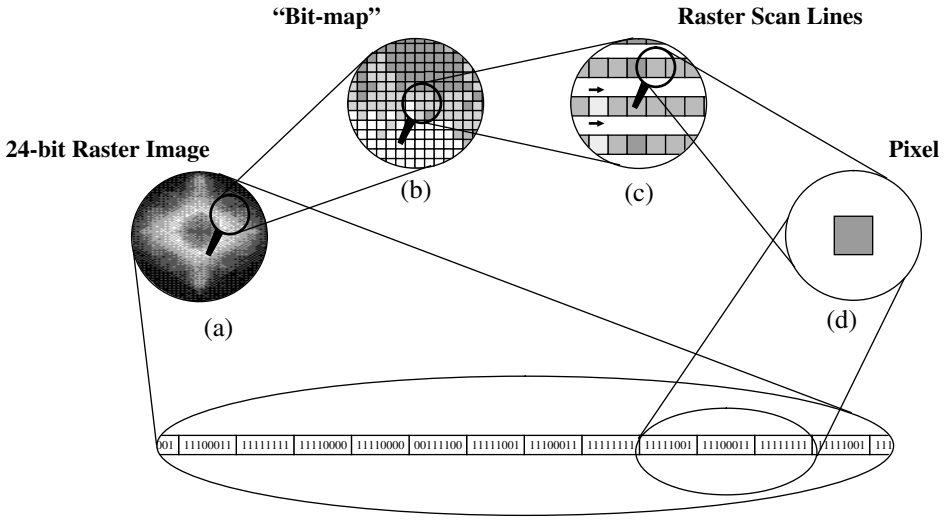
For every color appearing in an image, there are three 8-bit numbers in the color look-up table describing its appearance. Each 8-bit number or ***color component*** represents the amount of red, green, or blue used to create a particular color. Because each component is stored as an 8-bit number, the amount of red, green, or blue added to a color can range in value from 0 to 255. After a quick calculation, it's easy to see that there are 16.7 million (2^{24}) possible color combinations, but only 256 entries available in the color table. As a result, 256 colors out of a possible 16.7 million must be selected and assigned a position in the color look-up table. Selecting 256 colors for a color look-up table is also known as creating a palette.

In many ways, the ***palette model*** is the simplest data model in HDF. It provides a framework for storing color components as three 256-byte segments in a single 768-byte array. The palette model is most often used for working with multiple palettes, or palettes not specifically assigned to a raster image. To learn more about working with the palette model, consult the HDF User's Guide.

2.4 The 24-bit Raster Model

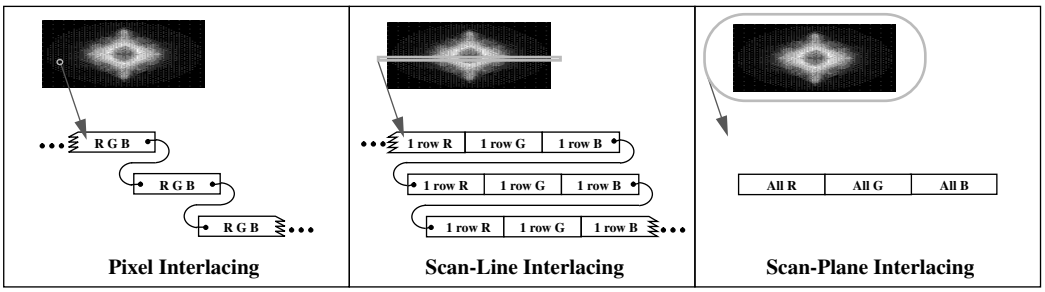
In terms of HDF, the definition of a 24-bit raster image is identical to that of an 8-bit raster image except that each pixel value is represented by three 8-bit numbers instead of one. More specifically, a ***24-bit raster image*** is any bit-mapped image whose rows are painted from left to right using three 8-bit numbers to represent each pixel value (See FIGURE 2.4a). Because each set of 8-bit numbers represents the RGB components for one pixel in the image, 24-bit images are not limited to the 256 color entries available in a color look-up table. The trade-off, however, is that 24-bit images occupy arrays three times larger than those used to store an equivalent set of 8-bit raster images.

FIGURE 2.4a **A 24-bit raster image(a) is a bit-map(b) whose rows are painted from left to right(c) using pixels(d) whose values are stored as three 8-bit numbers in a single stream of data(e).**



HDF supports three options for grouping the color components in 24-bit raster image data. These options consist of ***pixel interlacing***, ***scan-line interlacing***, and ***scan-plane interlacing*** (See FIGURE 2.4b). Storing the color components grouped by pixel, as in red-green-blue, red-green-blue, etc., is called pixel interlacing. Storing the color components by line, as in 1 row of red, 1 row of green, 1 row of blue, 1 row red, etc., is called scan-line interfacing. Finally, storing the color components grouped by color plane, as in the red components first, the green components second, and the blue components last, is called scan-plane interlacing. Unless otherwise specified, the HDF 24-bit raster model assumes that all 24-bit images are stored using pixel interlacing.

FIGURE 2.4b **HDF supports three different RGB interlacing schemes for 24-bit raster images.**

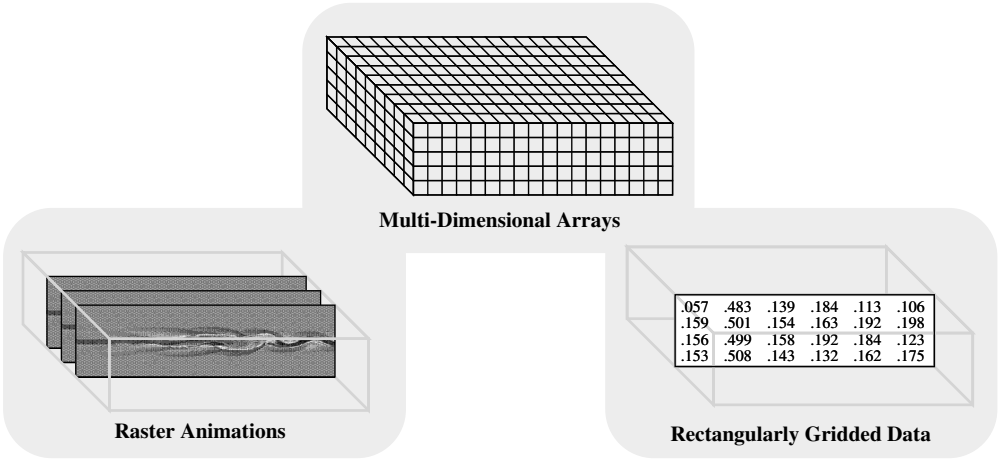


The ***24-bit raster model*** provides a framework for storing 24-bit raster images along with their image dimensions. Any 24-bit raster image accompanied by its image dimensions is known as a ***24-bit raster image set***. To learn more about 24-bit raster image sets, consult the HDF User's Guide.

2.5 The Scientific Data Model

Although any data contributing to a scientific investigation can be considered "scientific," HDF uses the phrase scientific data to refer to any data not specifically designed to be viewed as a raster image. Of course this does not preclude viewing scientific data as a raster image, it simply means that scientific data may be stored or viewed as something other than a two-dimensional array of 8, or 24-bit integers (See FIGURE 2.5a). In fact, HDF defines scientific data as any multi-dimensional array containing 8, 16, or 32-bit integers or 32- or 64-bit floating point numbers.

FIGURE 2.5a **Scientific data can be viewed in a variety of different ways.**



In addition to providing a framework for storing data of arbitrary dimensions and number type, the ***scientific data model*** also provides support for six individual ***attribute records***. Each attribute record is optional and describes one particular facet of the environment from which the scientific data was taken. The six predefined attribute records are described as follows:

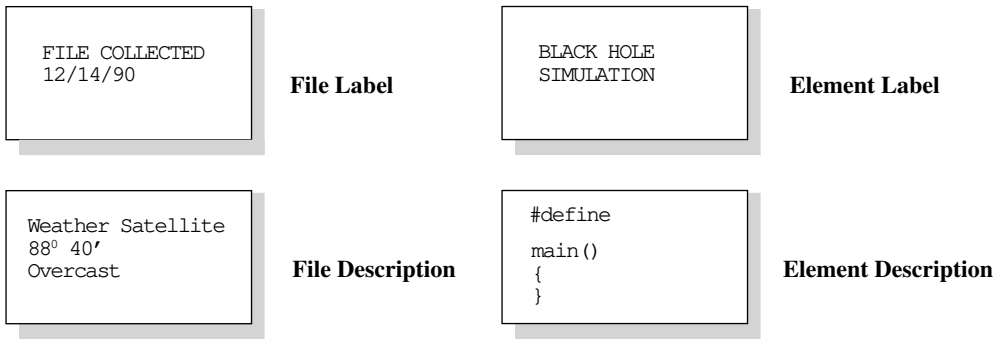
- **Coordinate System**: Identifies which coordinate system to use when displaying the data.
- **Formatting**: Specifies the format to use when displaying values for the data and its attributes.
- **Label**s: Contains a name for each independent variable and/or the data.
- **Max/Min Value**: Reports the maximum and minimum values found in the data.
- **Scale**: Describes the scale to use along each axis.
- **Units**: Identifies the unit associated with each dimension in the data.

In HDF, any multi-dimensional array accompanied by its dimension record and number type qualifies as a ***scientific data set***. Scientific data sets may also include attribute records, however, attribute records are optional and may be omitted. To learn more about scientific data sets, consult the HDF User's Guide.

2.6 The Annotation Model

When working with several different data types, it is often convenient to identify the contents of a file by adding a short text description or ***annotation***. Sometimes an annotation can serve as a short label for a file or data element, as in `COLLECTED 12/14/90` or `BLACK HOLE SIMULATION` (See FIGURE 2.6a). On other occasions, an annotation may require a more complex or lengthy description. For example, if the data collected on 12/14/90 originated as satellite data, the description might include the source of the data, pertinent environmental conditions, or other relevant information. Or in the case of the black hole simulation, the description might contain source code for the program that produced the simulation data.

FIGURE 2.6a **HDF can annotate both files and elements using labels or descriptions.**



Short annotations used to label files or data elements are called ***labels***, whereas long annotations are called ***descriptions***. Here the difference between short and long is not determined by byte length, but rather by the number of strings contained in the annotation. Labels are limited to one string containing any sequence of characters except zero, which will terminate the string. Descriptions, on the other hand, can contain any sequence of character data including zero, hence a description may contain any number of strings.

The HDF ***annotation model*** provides a framework for appending labels or descriptions to HDF files or the elements they contain. Although annotating a file is a rather simple operation, annotating a data element is bit more difficult. The difficulty lies in attaching the annotation to a specific piece of data in a file. To learn more about annotating HDF files and the elements they contain, consult the HDF User's Guide.

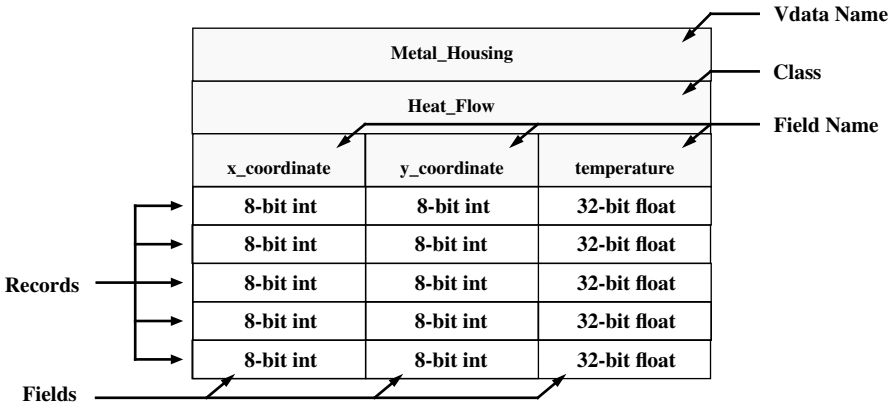
2.7 The Vdata Model

The HDF ***Vdata model*** provides a framework for storing customized tables in HDF files. Each table is comprised of a collection of similar records called ***Vdata records***, whose values are stored in fixed length fields. Each field can support its own number type, however, every Vdata record in a table must contain the same type of fields. Valid number types include 8-bit, 16-bit, and 32-bit signed and unsigned integers, and 32-bit and 64-bit floating point numbers.

Vdata tables use three kinds of identifying information: a name, a class, and a series of individual field names. A ***Vdata name*** is a label typically assigned to describe the origin or contents of a table. It often serves as a search key for files containing multiple tables. A ***Vdata class*** further distinguishes an individual table by identifying the meaning of its data. A class assignment often aids in determining how to interpret the fields in a record. Finally, ***Vdata field names*** identify the individual fields that make up a record.

Consider a collection of Vdata records containing two 8-bit fields followed by a 32-bit field (See FIGURE 2.7a) Assume that each record represents the location and temperature of a specific point on a heated metal housing. If each location is represented by an x, y, and temperature value, “x_coordinate,” “y_coordinate,” and “temperature” are certainly logical choices for their field names. “Heat_Flow” and “Metal_Housing” also represent appropriate possibilities for the Vdata class and name. Heat_Flow is a fairly accurate description of what type of records are stored in the table. Any software capable of recognizing Heat_Flow Vdata, can expect to encounter the location of each point followed by its temperature. Metal_Housing is a useful name for the Vdata table because it can serve as a search key in a file containing the heat flow tables for several different items.

FIGURE 2.7a **The constituent parts of a Vdata table.**



The HDF Vdata model provides a framework for storing Vdata records in customized tables. Since the Vdata interface is more complex than those of the other HDF data models, it is not covered in this document. To learn more about reading and writing Vdata, consult the HDF User's Guide.