* General Raster Images:  
  The GR Model

# Chapter Overview

This chapter provides functional descriptions of the GR Data Model, the GR implementation in the HDF library, and the HDF file structures employed.

* Section 8.2, "Images in an HDF File," describes the types of images that may be found in an HDF file.
* Section 8.3, "The GR Data Model," and Section 8.4, "Mapping between GR Data Model and HDF File Structures," describe the GR data model, including a rigorous UML representation, and the mapping of the model’s elements to HDF data structures.
* Section 8.5, "Modifying an RIG or RI8 Image via the GR Interface," discusses the interaction of the GR interface with older-style RIG and RI8 images.
* Section 8.6, "Backwards Compatibility when Creating New Images via the GR Interface," through Section 8.8, "Relationships among Main Data Structures," describe the GR implementation in the HDF library and the data structures employed.
* Section 8.9, "The Evolution of an HDF File in the GR Interface," then illustrates several steps in the evolution of the contents in an HDF file under the GR interface. At each step, the correspondence between the information as stored in memory and as represented in the file is described.

Many of the figures in this chapter employ UML notation (Unified Modeling Language notation) to show object relationships. See Section 7.2, "UML Notation and Object Symbols in HDF Data Model Descriptions."

## Images in an HDF File

An HDF file may contain many elements, including general raster images (GR data sets, the subject of this chapter) and older-style images, palettes, scientific data sets (SDSs), groups of HDF objects, annotations, etc. Figure 8a provides a high-level illustration of the elements of an HDF file.

* An HDF file may contain several objects and object collections

When a file is opened with the GR interface, all of the raster images in the file, including the older RI8, RIS8, and RIS24 images, become visible to the application, as illustrated in Figure 8ae below. Other objects in the file are unavailable through the GR interface; they can, however, be accessed through other interfaces, e.g., the H, V, and SD interfaces.



* An HDF file opened with the GR interface

As indicated in these figures, an HDF file may contain any of several styles of raster images; this is due to the history of HDF development and the need to maintain backwards compatibility. The older-style raster images, RIG and RI8, will occur in HDF files created with older versions of the HDF library. (See also Section 8.6, "Backwards Compatibility when Creating New Images via the GR Interface" regarding the current library’s ability to create these older-style images.) Figure 8af lists the properties of the three types of images, GR, RIG, and RI8, providing a tabulated comparison. The three following subsections describe these images in more detail.



* Three types of raster image



### GR data sets

The newest form of raster image in HDF is the general raster image. These images are represented by GR data sets and are referred to as such throughout this and other HDF documents. GR data sets were introduced at HDF Release 4.0.

GR data sets provide an extended color capability, global and local attributes, and special storage capabilities. The elements of a GR data set include the following HDF objects:

* Raster image data
* compressed image data (RLE or run length encoding, SKPHUFF or Skipping-Huffman, DEFLATE, and JPEG)
* special storage layout (compressed, chunked, compressed and chunked, or external)
* Image dimension
* Image attribute
* Palette
* Palette dimension

In the file, a GR data set consists of a Vgroup and several elements, as discussed in Section 8.4, "Mapping between GR Data Model and HDF File Structures," and illustrated in Figure 8am on page 93.

The GR data sets in a file constitute a GR collection, described in Section 8.3, "The GR Data Model."

GR data sets are created and manipulated via the GR interface (the GR API); see Section 8.9, "The Evolution of an HDF File in the GR Interface.". The GR interface also reads, and can manipulate, older-style raster images; see Section 8.5, "Modifying an RIG or RI8 Image via the GR Interface."

### RIG images (RIS8 and RIS24)

Raster image groups (RIGs), including RIS8 and RIS24 images, were the first HDF images to employ a grouping structure and provided the first 24-bit color image capability in HDF, while also providing extended compression capabilities. RIGs were the immediate predecessors to the GR approach and were introduced at HDF Release 2.0.

RIG images are represented by a raster image group (RIG) that contains pointers to other HDF objects. This type of raster image does not have attributes but does have all the other elements in the GR list above. Characteristics particular to RIGs are as follows:

* All RIG images are made up of 8-bit components.
* An RIS8 image is a 1-component, or 8-bit, RIG; an RIS24 image is a 3-component, or 24-bit, RIG.
* RIG compression modes are RLE (run-length encoding), IMCOMP, and JPEG.

Figure 8ag presents the file elements that make up an RIG image with a palette, which is optional.

* RIG with raster image and palette



An RIG is a tag/ref object and is fully described in Section 9.3.4, "Raster Image Tags," in Chapter , Tag Specifications. The DFTAG\_RI, DFTAG\_ID, DFTAG\_LUT, and DFTAG\_LD objects are fully described in the same chapter.

### RI8 images

The RI8 image is the original HDF 8-bit raster image and provides basic compression capabilities. RI8 images are characterized as follows:

* RI8 images employ no grouping structure.
* There are three compression modes for RI8 images:
* uncompressed images identified by the tag DFTAG\_RI8
* RLE-compressed images identified by the tag DFTAG\_CI8
* IMCOMP-compressed images identified by the tag DFTAG\_II8
* Image dimensions are identified by the tag DFTAG\_ID8.
* Palette dimensions are identified by the tag DFTAG\_IP8.

An RI8 image is a tag/ref object and is fully described in Section 9.3.9, "Obsolete Tags," in Chapter 9, Tag Specifications.

The ability of the current library to process RIG and RI8 images is intended only to support backward compatibility. The RIG and RI8 interfaces are both obsolete APIs and it is highly recommended that only the GR interface be used in new applications.

## The GR Data Model

This section provides a logical description of an HDF file containing GR images. A user’s view of the data model is presented in Section 8.3.1, "A Casual View," and Figure 8ah, "A sample user’s view of the GR model." The formal data model and a graphical representation are presented in Section 8.3.2, "The Formal GR Data Model," and Figure 8ai, "GR data model."

### A Casual View

From a user’s point of view, an HDF file containing GR data sets is structured as follows and as illustrated in Figure 8ah on page 90:

* The file contains GR data sets and optional global attributes.
* Every GR data set includes the following information:
* Name
* Number of components
* Dimension sizes (2 dimensions only)
* Pixel data type
* Image interlace mode (by pixel, line, or plane)
* Each GR data set may have the following associated elements and properties

|  |  |
| --- | --- |
| * Attribute(s) * A palette | * Data * Storage layout |



A palette is described by the following characteristics:

* Data type
* Number of entries
* Number of components
* Interlace mode

Global attributes, when present, are defined by the user, apply to all raster images in the file, and usually describe the intended usage of the GR data sets in the file. GR data set attributes, sometimes known as local attributes, are also optional, defined by the user, and describe only that data set.

GR data sets can have one of several storage layouts, as listed in Table 8a.

* GR storage layouts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GR data set | | | | |
| contiguous | special storage | | | |
| chunked | compressed | chunked and compressed | external |

* Contiguous storage is the default layout and requires no special storage tag.
* A sample user’s view of the GR model

For descriptions and definitions of the required and optional components that make up a general raster image, and of the GR interface routines provided by the HDF library to create and access GR data sets in the file, see Chapter 8, “General Raster Images (GR API),” in the HDF User’s Guide. For a complete description of palettes, see Chapter 9, “Palettes,” in the HDF User’s Guide.



### The Formal GR Data Model

The formal GR Data Model includes one type of object the user does not actually see, the GR collection. An HDF file may contain zero or one GR collection which may, in turn, contain zero or more GR data sets. The optional global attributes are actually associated with the GR collection.

A GR data set is an HDF data structure used to store a generalized raster image and the supporting metadata. Each GR data set may have zero or more associated attributes, sometimes referred to as local attributes.

The GR data sets and the associated objects (see Figure 8ai) can be accessed only through the GR interface.

* GR data model

The formal model is based on relationships among user-specified objects of the GR Data Model and the associated object attributes, as described in Figure 8aj.



* GR Data model objects

The GR interface provides routines to access the objects depicted in Section FIGURE 8ah, "A sample user’s view of the GR model," and Section FIGURE 8aj, "GR Data model objects." If an object is part of another object, it cannot be accessed by the GR interface without first accessing that other object; e.g., palette or attribute information can be accessed only after accessing the associated raster image.



## Mapping between GR Data Model and HDF File Structures

This section describes the mapping between the objects represented in the UML diagram in Figure 8ai, "GR data model," and the HDF objects in the file.

The illustrations in this section employ the symbols in Figure 8ak to identify file structures.

* File structure symbols

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vgroup |  |  | Other low-level HDF objects, usually identified by a tag/ref pair |  |
| Vdata |  |  | Abstract GR model object |  |

Elements of the GR data model map to HDF file objects as illustrated in Figure 8al

* Model-to-file mapping -- GR\_collection



A GR attribute is represented by a Vdata with one field. The field name is the name of the attribute. The field contains the value of the attribute; the number of records in the field corresponds to the number of attribute values. For example, the figure to the right represents an attribute named attribute\_name with the value abcd.



Figure 8am presents the file elements that make up an image, or GR data set, and the relationships among them as created by the GR interface.

* File structures representing a GR data set
* For any given image, the Vgroup may contain either



* raster image data, DFTAG\_RI or
* raster image data in a special storage format, indicated by an extended tag. Extended tags are described in Chapter 10, Extended Tags and Special Elements.
* The image dimension object, DFTAG\_ID, includes image dimension, interlace mode and compression information. Image compression may be RLE (run length encoding), SKPHUFF (Skipping-Huffman), DEFLATE, or JPEG.
* The GR data set Vgroup must have a class name of RI0.0. Should changes in the GR data structures ever become necessary, the class mechanism will enable the HDF library to manage evolving versions.

Figure 8an graphically presents the relationships among the elements of the formal GR data model. The GR collection is represented by a Vgroup whose members are the global attribute Vdata and the GR data set Vgroups. Each GR data set is represented by a Vgroup whose members are the image data and dimension objects, the palette objects, and the local attribute Vdata.

* File structures representing a GR collection



## Modifying an RIG or RI8 Image via the GR Interface

This section discusses the consequences of using the GR API to access and modify older-style RIG and RI8 images. This situation is likely to arise only when using the current version of the HDF library to edit a file that was created with an on older version.

Consider the file illustrated in Figure 8ao. This file contains one GR data set, one local attribute on that GR data set, one global attribute, one RIG image, and one palette on that RIG image.

* File with one GR data set and one RIG image

Now consider the use of the GR API to modify the RIG image.



First note that if the GR API modifies just the data of the RIG, e.g., the image or palette values or dimensions, but does not add an attribute, GR makes no changes to the file structure.

If an attribute is added, however, GR creates a Vgroup for a new GR data set, links the elements of the image (DFTAG\_RI or extended tag in the case of special storage, DFTAG\_ID, DFTAG\_LUT, and DFTAG\_LD) into that Vgroup, and adds the attribute Vdata.

The RIG group element (DFTAG\_RIG) is not linked into the GR data set Vgroup. The RIG image remains available via the older interfaces, though those interfaces will not show the attribute. Figure 8ap illustrates the structure of the file after an attribute has been added to the RIG image by means of the GR interface.

An RI8 image is incorporated into the GR collection under the same circumstances and in the same manner as the elements of an RIG image. The only difference is that there is not RIG object (DFTAG\_RIG) to consider.

When the GR interface is initiated, the information about the HDF file and its contents are mapped into memory and stored in the GR interface's main data structures, as discussed in Section 8.7, "Main Data Structures and their Relationships." These structures then maintain and update the information during processing of the application, and they are described in more details in the next section. When all processing is done, if the file contents have changed, the physical file will be updated with the information stored in the data structures.

* File of Figure 8ao after GR API has been used to add an attribute to the RIG image

## Backwards Compatibility when Creating New Images via the GR Interface



The HDF library makes extensive efforts to maintain backwards compatibility. When a new image is created via the GR interface, the library creates as many as possible of the following versions of the image:

* A GR data set is always created.
* An RIG is created for every image that meets the RIG criteria. For example, an RIG can be created for 1-component or 3-component images if the components are 8-bit integers and the compression mode is available for an RIG image. The images would be RIS8 or RIS24, respectively. If the image includes an attribute, that attribute will appear in the GR version of the image but will not be accessible in the RIG version.
* An RI8 image is created if the image meets the RI8 criteria. For example, an RI8 can be created for a 1-component, 8-bit image that uses a compression mode available for an RI8 image.

## Main Data Structures and their Relationships

This section provides the description of the main data structures used in the GR interface to store a GR data set's contents in memory. Figure 8aq lists these data structures and all their elements.

gr\_info\_t File information structure storing information about the HDF file.

ri\_info\_t Raster image information structure storing information about a raster image.

at\_info\_t Attribute information structure storing local and global attribute information.

dim\_info\_t Dimension information structure storing both image and palette dimension information.

These structures are somewhat self-described in Figure 8aq, except for some details too complex to present in the figure. The following subsections provide additional details about these structures. The last subsection in this section describes the relationships among the data structures.

* Main data structures in GR interface

### File Information Structure (gr\_info\_t)



The gr\_info\_t structure contains the information describing the HDF file whose identifier is stored in hdf\_file\_id (refer to Figure 8aq).

Additional details are as follows:

* gr\_ref is the reference number of the top level Vgroup in Figure 8an.
* grtree points to the tree whose nodes link to the raster image information structure describing an image in the file (see Figure 8at). Note that the images stored in this tree may include images read in from an existing file and images created in the application.
* gr\_count indicates the number of nodes in the tree grtree, i.e., the number of images currently stored in the file information structure.
* gr\_modified and gattr\_modified ensure that the file will be updated during GRend processing.
* gattree points to the tree whose nodes link to the attribute information structure which describes a global attribute in the file (see Figure 8at). Note that the attributes stored in this tree may include attributes read in from an existing file and attributes created in the application.
* gattr\_count indicates the number of nodes in the global attribute tree gattree, i.e., the number of global attributes currently stored in the file information structure.

### Raster Image Information Structure (ri\_info\_t)

The ri\_info\_t structure contains information describing a raster image.

When an existing file is opened, its contents are retrieved and stored in the data structures. The contents may include raster images, which may be of any type described in Section 8.2, "Images in an HDF File." The following table illustrates how different reference numbers in this structure are used to store the in-file representation of the three types of raster images. Notice that dim\_ref in the table belongs to the dimension information structure; however, because the dimension information structure is used by this image for both the image dimension and the image's palette dimension, it makes more sense to describe the dimensions' reference number here.

* Reference numbers and the in-file representation of raster images

|  |  |  |  |
| --- | --- | --- | --- |
|  | **GR data set** | **RIG raster image** | **Non-group raster image** |
| ri\_ref | Ref# of GR data set Vgroup | DFREF\_WILDCARD | DFREF\_WILDCARD |
| rig\_ref | aux\_ref? or DFREF\_WILDCARD | Ref# of RIG group | DFREF\_WILDCARD |
| img\_ref | Ref# of either the raster image data or the compressed image data | Ref# of either the raster image data or the compressed image data | Ref# of one of the following:   * 8-bit raster image * RLE compressed 8-bit raster image * IMCOMP compressed 8-bit raster image |
| lut\_ref | Ref# of the palette | Ref# of the palette | Ref# of one of the following:   * 8-bit palette * RLE compressed 8-bit palette * IMCOMP compressed 8-bit palette |
| img\_dim.dim\_ref | Ref# of the image dimension | Ref# of the image dimension | DFREF\_WILDCARD |
| lut\_dim.dim\_ref | Ref# of the palette dimension | Ref# of the palette dimension | DFREF\_WILDCARD |

Additional details are as follows:

* img\_dim is a structure describing the image dimension, as in Figure 8am and Figure 8an.
* lut\_dim is a structure describing the palette dimension in Figure 8am and Figure 8an.
* data\_modified, meta\_modified, and attr\_modified ensure that the file will be updated as necessary during the GRend processing.
* lattree points to the tree whose nodes link to the attribute information structure which describes an attribute of the image (see Figure 8au). Note that the attributes stored in this tree may include attributes read in from an existing file and attributes created in the application.
* lattr\_count indicates the number of nodes in the local attribute tree lattree, i.e., the number of image attributes currently stored in the file information structure.

### Attribute Information Structure (at\_info\_t)

The at\_info\_t structure is used to store the information describing a local or global attribute.

Additional details are as follows:

* ref is the reference number of the Vdata representing a global or local attribute in Figure 8an.
* new\_at ensures that an attribute that is newly created in an application is permanently recorded in the file before the file is closed. If this flag is set, GRend will add the tag/reference number pair of the Vdata that represents a local or global attribute to its RI Vgroup or the GR Vgroup, accordingly.

### Dimension Information Structure (dim\_info\_t)

The dim\_info\_t structure is used to store the information describing an image or palette dimension.

## Relationships among Main Data Structures

Figure 8ar provides a high-level illustration of the relationships among these data structures while Figure 8as, Figure 8at, and Figure 8au depict the relationships in more detail. As illustrated, the data structures TBBT\_TREE and TBBT\_NODE are widely used in the GR interface. TBBT\_TREE is a threaded, balanced, binary tree that is used to store different lists of objects and their information. Part of the definition of the tree can be found in Figure 8as. Basically, the tree is a structure that has a pointer, called root, pointing to another structure, TBBT\_NODE, which is a node of the tree. The main elements of TBBT\_NODE include two void pointers, data and key, and an array of three pointers that point to the parent, the left child, and the right child of the current node. The pointer data points to the data structure that is stored in this tree. The pointer key points to the value that is used to search for the data in the tree.

* High-level description of the relationships among the main data structures

Figure 8as shows a global tree gr\_tree that holds the GR file structure gr\_info\_t, which is used to store the file contents that are read into memory for processing or that are newly created and will be written to the file. The global tree gr\_tree is allocated when GRstart is first invoked in an application. A new structure of gr\_info\_t is also created and inserted into the tree at this time (routine New\_grfile). If GRstart is invoked more than once for a file in an application, then the global tree gr\_tree already exists and the current structure gr\_info\_t will be used (routine Get\_grfile). The key value used for searching in this tree is the HDF file identifier.



* The global GR tree

Figure 8at describes the elements of the GR file structure gr\_info\_t. This structure contains two TBBT\_TREE trees, grtree and gattree. The tree grtree contains the information for all the images in the file; thus, the pointer data in its nodes points to a raster image information structure, ri\_info\_t. Similarly, the tree gattree contains the information for all the global attributes in the file and its nodes point to the attribute information structure, at\_info\_t. If the file, which gr\_info\_t represents, has not been accessed in the current application, GRstart fills in the initial information of the GR file structure, which includes the creation of the two trees, grtree and gattree. GRstart then invokes GRIget\_image\_list to read in the file contents and store in the global tree gr\_tree as follows:



* For each of the global attributes, an attribute structure, at\_info\_t, is created and inserted into the attribute tree gattree, branching out from gr\_tree.
* For each of the raster images, a raster image structure, ri\_info\_t, is created and inserted into the grtree. Figure 8au illustrates the raster image structure and its main elements. These elements include two dimension information structures, dim\_info\_t, describing the image dimension and the image's palette dimension; a compression information structure, comp\_info, describing the image's compression; and a tree, TBBT\_TREE, holding all the attributes of the image.
* For each attribute of a raster image, an attribute structure, at\_info\_t, is created and inserted into the attribute tree lattree branching out from the raster image's structure.
* Illustration of data structure gr\_info\_t



* Illustration of data structure ri\_info\_t



## The Evolution of an HDF File in the GR Interface

This section illustrates several steps in the evolution of the contents in an HDF file under the GR interface. At each step, the correspondence between the information as stored in memory and as represented in the file is described.

* The file is created for access from the GR interface.
* Two raster images are created and written with data.
* Attributes are added to the file and to one of the raster images.
* A palette is added for one of the raster images.

The section also illustrates how the main GR structures represent the file elements in memory. The routines involved in constructing the file are described as necessary.

### Creating or Opening an HDF File

A typical HDF5 application calls the routine **Hopen** to create a new HDF file or to open an existing file.

Next, the routine **GRstart** is called to initiate the GR interface. GRstart does the following:

* Allocates the file information tree, gr\_tree. (Note that if GRstart is called more than once for the same HDF file, this tree will not be allocated again.)
* Initializes the atom groups for GR data sets (and older-style raster images).
* Retrieves the information of all contents in the file into the tree by invoking GRIget\_image\_list, which fills in gr\_tree with structures such as gr\_info\_t, ri\_info\_t, at\_info\_t, and dim\_info\_t.

At the end of GRstart, a newly created HDF file is represented in memory as shown in Figure 8av. Since there are neither images nor global attributes in the file, the roots of the image tree grtree and global attribute tree gattree point to NULL.

* Data structures of a newly created HDF file in memory

Note that the reference number gr\_ref in gr\_info\_t is DFREF\_WILDCARD at this time. That indicates that there is not yet a corresponding GR Vgroup in the file. This Vgroup is created during the GRend processing and gr\_ref will then have a valid reference number, which is that of the GR Vgroup and which will then be written into the file.



### Creating and Writing to a Raster Image

The routine **GRcreate** creates a raster image in the following steps:

* Creates an ri\_info\_t structure and fills it with initial information.
* Creates a Vgroup for this raster image, i.e., for this GR data set.
* Inserts the structure into the image tree (gr\_info\_t)grtree.

Figure 8aw illustrates the data structures after two raster images are created. The dashed boxes indicate the new data structures for the two new GR data sets. Notice that the local attribute trees lattree point to NULL indicating that the raster images have no attributes at this time. For the similar reason, the global tree gattree points to NULL. When GRend is invoked, the contents of the file are updated, causing these new images to be written to the file.

The file being assembled in these sections is illustrated in Figure 8ay, "File with two GR data sets, global attribute, local attribute, and image palette."

* Data structures storing two raster images

### Adding Attributes



The routine **GRsetattr** creates an attribute for a file or for a raster image in the following steps:

* If the attribute already exists in the file, then simply updates it, although, the number type cannot be changed
* If the attribute's data is small enough to be cached, keeps the data in memory where specified by (at\_info\_t)data.
* Otherwise, writes the data to the attribute Vdata on disk.
* If the attribute is new, the following actions are performed:
* Creates the attribute structure at\_info\_t and stores the attribute information.
* If the attribute's data is small enough to be cached, keeps the data in memory where specified by (at\_info\_t)data.
* Otherwise, writes the data to the attribute Vdata on disk.
* Adds the attribute structure to the attribute tree, which can be either the global attribute tree (gr\_info\_t)gattree or the local attribute tree (ri\_info\_t)lattree.

Figure 8ax shows the memory data structures with two raster images, one file attribute, and one local attribute. An at\_info\_t structure is also added to the global attribute tree for the new file attribute. When GRend is invoked, the contents of the file are updated, causing these attributes be written to the file.

* Data structures after adding two attributes

### Adding Palettes



The routine GRwritelut writes the palette of a raster image in the following steps:

* Makes certain that only standard palettes are written.
* If the palette object already exists for the image, simply writes the palette data to the file.
* Otherwise, creates the palette dimension, initializes it, then creates the palette object and writes the palette data to the file.

There are no structural changes in the data structures. The palette dimension is filled with initial information and the palette object's tag and reference number are stored in the raster image information structure. Figure 8ay shows the representation of the file with the new palette object.

* File with two GR data sets, global attribute, local attribute, and image palette

### Opening an Existing File



When the HDF file already exists and is opened for processing, the data structure gr\_info\_t, which includes the part enclosed in the dotted box in Figure 8av, is filled with the file contents. For example, Figure 8aw shows the in-memory storage of the file that is represented in Figure 8ay. The routine GRIget\_image\_list is responsible for retrieving the file contents and storing them in memory. The retrieval process is carried out as follows:

* Collect all the raster images in the file, including all three types.
* Collect all the global attributes and, for each attribute, create an at\_info\_t structure and store it on the global attribute tree gattree, branched out from the gr\_info\_t structure.
* Eliminate any duplications among the raster images found.
* For each raster image, the following actions are performed:
* Create an ri\_info\_t structure and fill it with information about the raster image.
* If any raster image has attributes, for each attribute, create an at\_info\_t structure and store it on the local attribute tree lattree, branched out from the ri\_info\_t structure.
* Store image dimension information in the structure img\_dim of the ri\_info\_t structure.
* Store palette dimension information in the structure lut\_dim of the ri\_info\_t structure.
* Finally, store the ri\_info\_t structure for this raster image on the image tree grtree, branched out from the gr\_info\_t structure.