DFR8addimage

```
intn DFR8addimage(char *filename, VOIDP image, int32 width, int32 height,
          uint16 compress)
```

filename	IN:	Name of HDF file
image	IN:	Array holding image to add to file
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image
compress	IN:	Type of compression to use, if any

Purpose **DFR8addimage** appends the RIS8 for the image to the file.

Return value Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.

Description To append to the file the RIS8 for the image. In all other respects, **DFR8addimage** is functionally equivalent to **DFR8putimage**.

FORTRAN

```
integer function d8aimg(filename, image, width, height,
compress)
    character*(*)   filename
    <valid type>    image(*)
    integer*4       width, height
    integer*2       compress
```

DFR8getdims

d8gdims

DFR8getdims

```
intn DFR8getdims(char *filename, int32 *width, int32 *height, intn *ispalette)
```

filename	IN:	Name of HDF file containing the 8-bit raster image(s)
width	OUT:	Number of columns in the next image in the file
height	OUT:	Number of rows in the next image in the file
ispalette	OUT:	Returns "1" if there is a palette, otherwise returns "0"

Purpose

To open the file, find the next image, retrieve the dimensions of the image, and determine whether there is a palette associated with the image.

Return value

Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.

Description

`DFR8getdims` retrieves the dimensions of the image and indicates whether a palette is associated and stored with the image. If the file is being opened for the first time, **`DFR8getdims`** returns information about the first image in the file. If an image has already been read, **`DFR8getdims`** finds the next image. Thus, images are read in the same order in which they were written to the file.

Normally, **`DFR8getdims`** is called before **`DFR8getimage`** so that if necessary, space allocations for the image and palette can be checked, and the dimensions can be verified. If this information is already known, **`DFR8getdims`** need not be called.

FORTRAN

```
integer function d8gdims(filename, width, height, ispalette)
    character*(*)  filename
    integer*4      width, height
    integer        ispalette
```

DFR8getimage

```
intn DFR8getimage(char *filename, uint8 *image, int32 width, int32 height,
          uint8 *palette)
```

filename	IN:	Name of file with RIS8 image
image	OUT:	Array that will hold image
width	IN:	Width of buffer allocated to store image
height	IN:	Height of buffer allocated to store image
palette	OUT:	Palette to go with image

Purpose — To retrieve the image and its palette, if it is present, and store them in the specified arrays.

Return value — Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.

Description — If palette is `NULL`, no palette is loaded, even if there is one stored with the image. If the image in the file is compressed, **`DFR8getimage`** automatically decompresses it. If **`DFR8getdims`** has not been called, **`DFR8getimage`** finds the next image in the same way that **`DFR8getdims`** does.

The variables `width` and `height` give the number of columns and rows, respectively, in the array which you've allocated in memory to store the image. The image may actually be smaller than the allocated space.

The order in which you declare dimensions is different between C and FORTRAN. Ordering varies because FORTRAN arrays are stored in column-major order, while C arrays are stored in row-major order. (Row-major order implies that the horizontal coordinate varies fastest). When **`d8gimg`** reads an image from a file, it assumes row-major order. The FORTRAN declaration that causes an image to be stored in this way must have the width as its first dimension and the height as its second dimension. To take this into account as you read image in your program, you need to read in the image “on its side.”

FORTRAN

```
integer function d8gimg(filename, image, width, height,
palette)
    character*(*)  filename
    character*1    image(*), palette(*)
    integer*4      width, height
```

DFR8lastref

d8lref

DFR8lastref

```
uint16 DFR8lastref()
```

Purpose	To get last reference number written or read for an RIS8.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	This routine is primarily used for attaching annotations to images and adding images to Vgroups. **`DFR8lastref`** returns the reference number of last raster image set read or written.
FORTRAN	`integer function d8lref()`

DFR8nimages

```
intn DFR8nimages(char *filename)
```

filename	IN:	Name of HDF file

Purpose — To count the number of 8-bit raster images contained in an HDF file.

Return value — Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.

Description — **`DFR8nimages`** counts the number of 8-bit images stored in the file.

FORTRAN

```
integer function d8nims(filename)
    character*(*)  filename
```

DFR8putimage

d8pimg

DFR8putimage

```
intn DFR8putimage(char *filename, VOIDP image, int32 width, int32 height,
          uint16 compress)
```

filename	IN:	Name of file to store the raster image in
image	IN:	Array with image to put in file
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image
compress	IN:	Type of compression to use: 0=none, DFTAG_RLE, or DFTAG_IMCOMP

Purpose

To write out the RIS8 for the image as the first image in the file, overwriting any information that was previously in the file.

Return value

Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description

DFR8putimage writes out the raster image set for the image as the first image in the file. The argument compress identifies the scheme to be used for compressing the data, if any. If IMCOMP compression is used, the image must include a palette.

DFR8putimage overwrites any information that exists in the HDF file. To write an image to a file by appending it, rather than overwriting it, use **DFR8addimage**.

In FORTRAN, the dimensions of the array image must be the same as the dimensions of the image itself.

The order in which you declare dimensions is different between C and FORTRAN. Ordering varies because FORTRAN arrays are stored in column-major order, while C arrays are stored in row-major order. (Row-major order implies that the horizontal coordinate varies fastest). When **DFR8putimage** writes an image to a file, it assumes row-major order. The FORTRAN declaration that causes an image to be stored in this way must have the width as its first dimension and the height as its second dimension, the reverse of the way it is done in C. To take this into account as you build your image in your FORTRAN program, you need to build the image "on its side."

FORTRAN

```
integer function d8pimg(filename, image, width, height,
compress)
    character*(*)  filename
    <valid type>   image(*)
    integer*4      width, height
    integer*2      compress
```

DFR8readref

```
intn DFR8readref(char *filename, uint16 ref)
```

filename	IN:	Name of HDF file containing image
ref	IN:	Reference number for next **DFR8getimage**

Purpose

To specify the reference number of the image to be read when **DFR8getimage** is next called.

Return value

Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description

DFR8readref sets the reference number of the image to get next. This routine is most likely to be used in conjunction with **DFANgetlablist**, which returns a list of labels for a given tag together with their reference numbers. It provides, in a sense, a random access to images.

There is no guarantee that reference numbers appear in sequence in an HDF file; therefore, it is not safe to assume that a reference number is the sequence number for an image.

FORTRAN

```
integer function d8rref(filename, ref)
    character*(*)  filename
    integer*2      ref
```

DFR8restart

d8first

DFR8restart

```
intn DFR8restart()
```

Purpose	To cause the next get to read from the first RIS8 in the file, rather than the RIS8 following the one that was most recently read.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	**`DFR8restart`** causes the next get command to read from the first raster image set in the file
FORTRAN	`integer function d8first()`

DFR8setcompress

```
intn DFR8setcompress(int32 type, comp_info *cinfo)
```

type	IN:	Type of compression
cinfo	IN:	Pointer to compression information structure

Purpose Set the type of compression to use when writing the next 8-bit raster image.

Return value Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description This routines provides a method for compressing the next raster image written. The *type* can be one of the following values: COMP_NONE, COMP_JPEG, COMP_RLE, COMP_IMCOMP. COMP_NONE is the default for storing images if this routine is not called, therefore images are not compressed by default. COMP_JPEG compresses images with a JPEG algorithm, which is a lossy method. COMP_RLE uses lossless run-length encoding to store the image. COMP_IMCOMP uses a lossy compression algorithm called IMCOMP, and is included for backward compatibility only.

The *cinfo* parameter is a pointer to a union defined as:

```
union {
    struct {
        intn quality;
        intn force_baseline;
    } jpeg;
} comp_info;
```

This union is defined to provide future expansion, but is currently only used by the COMP_JPEG compression type. A pointer to a valid comp_info union is required for all compression types other than COMP_JPEG, but the values in the union are not used. The comp_info union is declared in the header file hdf.h and is shown here for informative purposes only, it should not be re-declared in a user program.

For COMP_JPEG compression, the *quality* member of the *jpeg* structure must be set to the quality of the stored image. This number can vary from 100, the best quality, to 0, terrible quality. All images stored with COMP_JPEG compression are stored in a lossy manner, even images stored with a quality of 100. The ratio of size to perceived image quality varies from image to image, some experimentation may be required to determine an acceptable *quality* factor for a given application. The *force_baseline* parameter determines whether the quantization tables used during compression are forced to the range 0-255. The *force_baseline* parameter should normally be set to 1 (forcing baseline results), unless special applications require non-baseline images to be used.

FORTRAN None

DFR8setpalette

d8spal

DFR8setpalette

```
intn DFR8setpalette(uint8 *palette)
```

palette	IN:	Palette to go with the image

Purpose	Indicate which palette, if any, is to be used for subsequent image sets.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	Sets the default palette to be used for subsequent images. The specified palette continues as the default palette until it is changed by a new call to **`DFR8setpalette`**.
FORTRAN	`integer function d8spal(`*`palette`*`)` `  character*1 `*`palette`*`(*)`

DFR8writeref

```
intn DFR8writeref(char *filename, uint16 ref)
```

filename	IN:	HDF file containing image
ref	IN:	Reference number for next **DFR8putimage** or **DFR8addimage**

Purpose — To specify the reference number of the image to be written when **DFR8addimage** or **DFR8putimage** is next called.

Return value — Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description — **DFR8writeref** assigns the specified reference number to the next image written to file. It is unlikely that you will need this routine, but if you do, use it with caution. There is no guarantee that reference numbers appear in sequence in an HDF file; therefore, it is not safe to assume that a reference number is the sequence number for an image. In addition, using an existing reference number will overwrite the existing 8-bit raster image data.

FORTRAN

```
integer function d8wref(filename, ref)
    character*(*)  filename
    integer*2      ref
```