DF24addimage

```
intn DF24addimage(char *filename, VOIDP image, int32 width, int32 height)
```

filename	IN:	Name of HDF file
image	IN:	Pointer to the image array
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image

Purpose — To write out a 24-bit image.

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

Description — **`DF24addimage`** appends a 24-bit raster image set for the image to the file. Array *`image`* is assumed to be `width` x `height` x 3 bytes. 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 last coordinate varies fastest).

When **`DF24putimage`** 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. 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 d2aimg(filename, image, width, height)
    character*(*)  filename
    <valid type>   image(*)
    integer*4      width, height
```

DF24getdims

d2gdims

DF24getdims

```
intn DF24getdims(char *filename, int32 *width, int32 *height, intn *il)
```

filename	IN:	Name of HDF file
width	OUT:	For returning width of image
height	OUT:	For returning height of image
il	OUT:	For returning interlace of image in file

Purpose

To get dimensions and interlace storage scheme of next image RIS.

Return value

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

Description

`DF24getdims` retrieves the dimensions and interlace of the image. If the file is being opened for the first time, **`DF24getdims`** returns information about the first image in the file. If an image has already been read, **`DF24getdims`** finds the next image. Thus, images are read in the same order in which they were written to the file.

If you know the dimensions and interlace of the image beforehand, there is no need to call **`DF24getdims`**. Simply allocate arrays with the proper dimensions for the image and let **`DF24getimage`** read in the images. If, however, you do not know the values of *`width`* and *`height`*, you must call **`DF24getdims`** to get them and then use them to determine the right amount of space needed for the array image.

Successive calls to **`DF24getdims`** and **`DF24getimage`**, respectively, retrieve all of the images in the file in the sequence in which they were written.

Interlace codes: `0` = pixel interlacing; `1` = scan-line interlacing; `2` = scan-plane interlacing.

FORTRAN

```
integer function d2gdims(filename, width, height, il)
   character*(*)  filename
   integer*4      width, height
   integer        il
```

DF24getimage

```
intn DF24getimage(char *filename, VOIDP image, int32 width, int32 height)
```

filename	IN:	Name of HDF file
image	OUT:	Pointer to space to return image
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image

Purpose — To get image from next 24-bit raster image set.

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

Description — **`DF24getimage`** retrieves the image and stores it in an array. If **`DFR24getdims`** has not been called, **`DFR24getimage`** finds the next image in the same way that **`DFR24getdims`** does.

The amount of space allocated for the image should be `width` x `height` x 3 bytes.

To specify that the next call to **`DF24getimage`** should read the raster image using an interlace other than the interlace used to store the image in the file, make a call to **`DF24reqil`**.

FORTRAN

```
integer function d2gimg(filename, image, width, height)
   character*(*)  filename
   <valid type>   image(*)
   integer*4      width, height
```

DF24lastref

d2lref

DF24lastref

```
uint16 DF24lastref()
```

Purpose	To get last reference number written or read for a 24-bit raster image set.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	This routine is primarily used for attaching annotations to 24-bit images and adding 24-bit images to Vgroups. **`DF24lastref`** returns the reference number of last 24-bit raster image read or written.
FORTRAN	`integer function d2lref()`

DF24nimages

```
intn DF24nimages(char *filename)
```

filename	IN:	Name of the HDF file

Purpose	To count the number of 24-bit raster images contained in an HDF file.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	**`DF24nimages`** counts the number of 24-bit images stored in the file.
FORTRAN	`integer function d2nims(filename)` `  character*(*)  filename`

DF24putimage

d2pimg

DF24putimage

```
intn DF24putimage(char *filename, VOIDP image, int32 width, int32 height)
```

filename	IN:	Name of HDF file
image	IN:	Pointer to the image array
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image

Purpose
To write out a 24-bit image as the first image in the file, overwriting any information previously stored in the file.

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

Description
`DF24putimage` writes a 24-bit raster image to a file. The array *`image`* is assumed to be `width` x `height` x 3 bytes. **`DF24putimage`** overwrites any information that exists in the HDF file. To append a new image to a file instead of overwriting an existing file, use **`DF24addimage`**.

FORTRAN

```
integer function d2pimg(filename, image, width, height)
   character*(*)   filename
   <valid type>    image(*)
   integer*4       width, height
```

DF24readref

```
intn DF24readref(char *filename, uint16 ref)
```

filename	IN:	Name of HDF file containing image
ref	IN:	Reference number for next **DF24getimage**

Purpose

To specify the reference number of the image to be read when **DF24getimage** is next called.

Return value

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

Description

DF24readref sets the reference number for the image to get next. You will most likely use this routine 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 d2rref(filename, ref)
    character*(*)  filename
    integer*2      ref
```

DF24reqil

d2reqil

DF24reqil

```
intn DF24reqil(intn il)
```

il	IN:	Interlace to get next image with

Purpose — To cause next **DF24getimage** to store image in memory with the specified interlace.

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

Description — **DF24reqil** specifies an interlace to be used in place of the interlace indicated in the file when the next raster image is read. Regardless of what interlace scheme is used to store the image, **DF24reqil** causes the image to be loaded into memory and be interlaced according to the specification of *il*.

Since a call to **DF24reqil** may require a substantial reordering of the data, I/O performance could be adversely affected; e.g. it could result in much slower I/O performance than would be achieved if no change in interlace were requested.

Interlace codes: `0` = pixel interlacing; `1` = scan-line interlacing; `2` = scan-plane interlacing.

FORTRAN

```
integer function d2reqil(il)
    integer il
```

DF24restart

```
intn DF24restart()
```

Purpose	To cause the next get to read from the first RIS24 in the file, rather than the RIS24 following the one that was most recently read.
Return value	Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.
Description	**`DF24restart`** causes the next get command to read from the first 24-bit raster image set in the file, rather than the next one.
FORTRAN	`integer function d2first()`

DF24setcompress

```
intn DF24setcompress(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 24-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

DF24setdims

```
intn DF24setdims(int32 width, int32 height)
```

width	IN:	Number of columns in the image
height	IN:	Number or rows in the image

Purpose — Set the dimensions for the next image to be written to a file.

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

Description — Sets the dimensions of the next image to be written to the HDF file.

FORTRAN

```
integer function d2sdims(width, height)
    integer*4 width, height
```

DF24setil

d2setil

DF24setil

```
intn DF24setil(intn il)
```

il	IN:	Interlace of image

Purpose — To set interlace scheme to be used on subsequent writes.

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

Description — **`DF24setil`** sets the interlace to be used when writing out the raster image set for the 24-bit image. If **`DF24setil`** is not called, the interlace code is assumed to be 0.

Interlace codes: `0` = pixel interlacing; `1` = scan-line interlacing; `2` = scan-plane interlacing.

FORTRAN

```
integer function d2setil(il)
    integer il
```

DF24writeref

```
intn DF24writeref(char *filename, uint16 ref)
```

filename	IN:	HDF file containing image
ref	IN:	Reference number for next **DF24putimage** or **DF24addimage**

Purpose — To specify the reference number of the image to be written when **DF24addimage** or **DF24putimage** is next called.

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

Description — **DF24writeref** sets the reference number of the image to write next. 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.

FORTRAN

```
integer function d2wref(filename, ref)
    character*(*)  filename
    integer*2      ref
```