Chapter 10

Annotations (DFAN and AN API)

10.1 Chapter Overview

The HDF annotation interface supports the appending of labels and descriptions to HDF files and the data objects they contain. This chapter explains the methods used to read and write file and object annotations. It also explains the alternate multifile annotation interface introduced in version 4.0.

10.2 The Annotation Data Model

When working with different data types, it is often convenient to identify the contents of a file by adding a short text description or ***annotation***. An annotation serves as the label for a file or data element, as in "COLLECTED 12/14/90" or "BLACK HOLE SIMULATION". For example, if the data originated as satellite data, the description might include the source of the data, pertinent environmental conditions, or other relevant information. In the case of a hypothetical black hole simulation, the description might contain source code for the program that produced the data.

As they are implemented as variable-length strings, HDF annotations are designed to accommodate a wide variety of information including titles, comments, variable names, parameters, formulas, and source code. In fact, HDF annotations can encompass any textual information regarding the collection, meaning, or intended use of the data.

10.2.1 Labels and Descriptions

Annotations come in two forms; ***labels*** and ***descriptions***. Labels are short annotations used for assigning things like titles or time stamps to a file or its data objects. Longer annotations are called descriptions, and they typically contain more extensive information such as a source code module or mathematical formulae.

Labels are defined as a null-terminated string of characters except `NULL`. Descriptions may contain any sequence of ASCII characters including `NULL`.

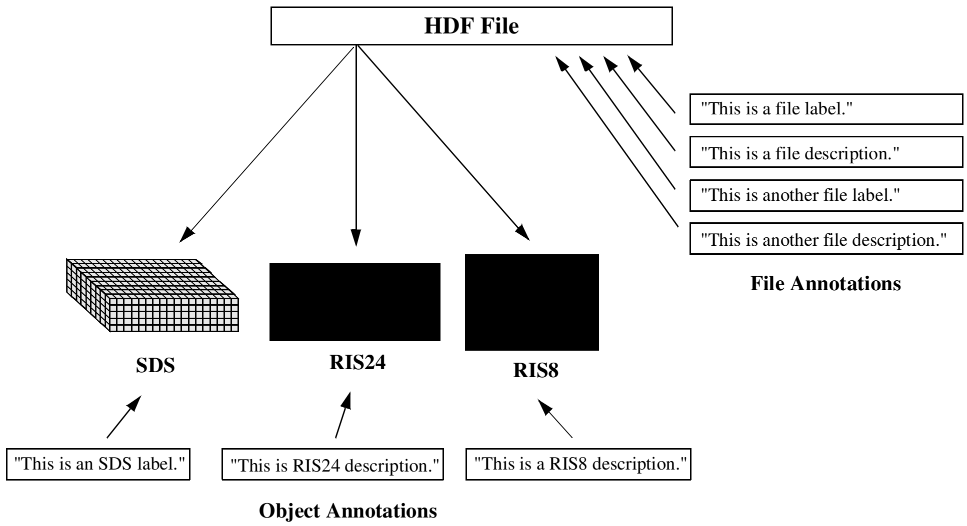
In addition to the distinction made between labels and descriptions, HDF distinguishes between ***file annotations*** and ***object annotations***.

10.2.2 File Annotations

File annotations are text strings which describe the origin, meaning, or intended use of its data. Any HDF file can be annotated with a label, description, or combination of both. (See Figure 10a.) The number of labels or descriptions an HDF file may contain is limited to the maximum number of tag/reference number pairs. File annotations may be assigned in any order and at any time after a file is created.

FIGURE 10a

File and Object Annotations



Although it may seem convenient to use a file annotation to describe one data object in a file, this practice is not recommended because any additional objects added to the file will obscure the implicit link between the original object and its annotation.

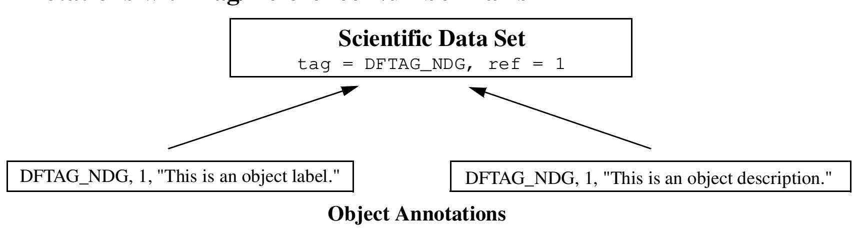
10.2.3 Object Annotations

Object annotations are text strings assigned to individual data objects to explain their origin, meaning, or intended use. Because object annotations are assigned to individual objects, their use requires an understanding of HDF tags and reference numbers.

The annotation interface takes advantage of this identification scheme by including the object's tag/reference number pair with the text of the annotation. Consider a scientific data set identified by the tag `DFTAG_NDG` and the reference number `1`. (See Figure 10b.) All object annotations assigned to this particular data set must be prefaced with the tag `DFTAG_NDG` followed by the reference number `1`.

FIGURE 10b

Object Annotations with Tag/Reference Number Pairs



10.3 The Single-File Annotation Interface

The functions and routines that comprise the single-file annotation interface have names that begin with the string "DFAN" and the equivalent Fortran-77 routine names are prefaced by "da".This interface is the older annotation interface and only supports annotation access within one particular HDF file. It doesn't support the concept of an annotation identifier used in the newer multifile interface, therefore annotations created with the multifile interface cannot be accessed or manipulated with single-file interface functions. The multifile interface is described in Section 10.9 on page 264.

DFANaddfid, **DFANgetfid**, **DFANaddfds**, and **DFANgetfds** are the functions used for annotating files and are sufficient for most read and write operations.

10.3.1 Single-File Annotation Library Routines

These functions are divided into the following categories:

- ***Write functions*** assign a file or object annotation.
- ***Read functions*** retrieve a file or object annotation.
- ***General inquiry functions*** return a list of all labels and reference numbers.

The DFAN function calls are defined in the following table and on their respective pages in the *HDF Reference Guide*.

TABLE 10A **DFAN Library Routines**

Purpose	Functions		Description
	C	Fortran-77	
Write	DFANaddfid	daafid	Assigns a file label to a specific file.
	DFANaddfds	daafds	Assigns a file description to a specific file.
	DFANputlabel	daplab	Assigns an object label to a specific data object.
	DFANputdesc	dapdesc	Assigns an object description to a specific data object.
Read	DFANgetfidlen	dagfidl	Returns the length of a file label.
	DFANgetfid	dagfid	Reads the text of a file label.
	DFANgetfdslen	dagfdsl	Returns the length of a file description.
	DFANgetfds	dagfds	Reads the text of a file description.
	DFANgetlablen	dagllen	Returns the length of an object label.
	DFANgetlabel	daglab	Reads the text of an object label.
	DFANgetdesclen	dagdlen	Returns the length of an object description.
	DFANgetdesc	dagdesc	Reads the text of an object description.
General Inquiry	DFANlablist	dallist	Gets a list of all the labels in a file for a particular tag.
	DFANlastref	dalref	Returns the reference number of the last annotation accessed.
Maintenance	DFANclear	None	Clear the internal tables and strcutures used by the DFAN interface.

10.3.2 Tags in the Annotation Interface

See Table 10B below for a list of the annotation tags defined in HDF versions 2.0, 3.0 and 4.0. Newly-defined tags names in each version are bolded. For a more complete list of tags consult the *HDF Specification and Developer's Guide*.

TABLE 10B **List of Annotation Interface Tags in HDF Versions 2.0, 3.0 and 4.0**

Interface	Data Object	Tag Name		
		v2.0	v3.0	v4.0

DFR8	Raster Image: 8-bit (uncompressed)	DFTAG_RI8	DFTAG_RI	DFTAG_RI
	Compressed Image: 8-bit	DFTAG_CI8	DFTAG_CI	DFTAG_CI
	Image Dimension: 8-bit	DFTAG_ID8	DFTAG_ID	DFTAG_ID
	Image Palette: 8-bit	DFTAG_IP8	DFTAG_LUT	DFTAG_LUT
DF24	Raster Image Group	None	DFTAG_RIG	DFTAG_RIG
	Raster Image (uncompressed)	None	DFTAG_RI	DFTAG_RI
	Compressed Image	None	DFTAG_CI	DFTAG_CI
	Image Dimension	None	DFTAG_ID	DFTAG_ID
DFP	Color Look-up Table	DFTAG_LUT	DFTAG_LUT	DFTAG_LUT
DFSD	Scientific Data Group	DFTAG_SDG	DFTAG_SDG	DFTAG_NDG
	Scientific Data	DFTAG_SD	DFTAG_SD	DFTAG_SD
	Scientific Data Dimension	DFTAG_SDD	DFTAG_SDD	DFTAG_SDD
	Scientific Data Scale Attribute	DFTAG_SDS	DFTAG_SDS	DFTAG_SDS
	Scientific Data Label Attribute	DFTAG_SDL	DFTAG_SDL	DFTAG_SDL
	Scientific Data Unit Attribute	DFTAG_SDU	DFTAG_SDU	DFTAG_SDU
	Scientific Data Format Attribute	DFTAG_SDF	DFTAG_SDF	DFTAG_SDF
	Scientific Data Max/Min Attribute	DFTAG_SDM	DFTAG_SDM	DFTAG_SDM
	Scientific Data Coordinates Attribute	DFTAG_SDC	DFTAG_SDC	DFTAG_SDC
DFAN	File Identifier	DFTAG_FID	DFTAG_FID	DFTAG_FID
	File Descriptor	DFTAG_FD	DFTAG_FD	DFTAG_FD
	Data Identifier Label	DFTAG_DIL	DFTAG_DIL	DFTAG_DIL
	Data Identifier Annotation	DFTAG_DIA	DFTAG_DIA	DFTAG_DIA
Vdata	Vdata Storage	DFTAG_VS	DFTAG_VS	DFTAG_VS
Vgroups	Vgroup Storage	DFTAG_VG	DFTAG_VG	DFTAG_VG

10.4 Programming Model for the Annotation Interface

There are two general programming models for the DFAN interface. The first programming model addresses file annotation, and the second addresses object annotation. In the case of file annotations, the DFAN interface relies on the calling program to initiate and terminate access to files. This approach necessitates the following programming model:

1. Open the file.
2. Perform the desired file annotation operation.
3. Close the file.

The object annotation programming model is a simplified version of the file annotation programming model:

1. Perform the desired object annotation operation.

Essentially, the difference between the two models is that file annotations require **Hopen** and **Hclose** to open and close the target files whereas object annotations do not.

10.5 Writing Annotations

The DFAN interface supports writes to file labels, file descriptions, object labels and object descriptions.

10.5.1 Assigning a File Label: DFANaddfid

To write a file label, the calling program must call **DFANaddfid**:

C: `status = DFANaddfid(file_id, label);`

FORTRAN: `status = daafid(file_id, label)`

DFANaddfid has two parameters: `file_id` and `label`. The `file_id` parameter contains the file identifier for the file to be annotated and the `label` parameter contains the annotation string. The label array must be null-terminated. In the Fortran-77 version, the length of the label should be the length of the label array as in Fortran-77 string lengths are assumed to be the declared length of the array that holds the string.

The parameters of **DFANaddfid** is further defined below. (See Table 10F on page 259.)

10.5.2 Assigning a File Description: DFANaddfds

To write a file description, the calling program must call **DFANaddfds**:

C: `status = DFANaddfds(file_id, description, desc_length);`

FORTRAN: `status = daafds(file_id, description, desc_length)`

DFANaddfds has three parameters: `file_id`, `description`, and `desc_length`. The `file_id` parameter contains the file identifier and the `description` parameter contains the label string. The parameter description can contain any sequence of ASCII characters and is not limited to a single string. The `desc_length` parameter specifies the length of the annotation.

The parameters of **DFANaddfds** are defined in the following table.

TABLE 10C **DFANaddfid and DFANaddfds Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANaddfid (daafid)	file_id	int32	integer	File identifier.
	label	char *	character* (*)	File label string.
DFANaddfds (daafds)	file_id	int32	integer	File identifier.
	description	char *	character* (*)	File description string.
	desc_length	int32	integer	Length of the description in bytes.

EXAMPLE 3. **Writing a File Label and a File Description**

The following examples add a file label and description to the file named "Example1.hdf". Notice that after the file is opened, the `file_id` may be used to add any combination of file annotations before the file is closed.

C:

```
#include "hdf.h"
```

```
main( )
{

int32 file_id;
intn status;
static char file_label[] = "This is a file label.";
static char file_desc[] = "This is a file description.";

/* Open the HDF file to write the annotations. */
file_id = Hopen("Example1.hdf", DFACC_CREATE, 0);

/* Write the label to the file. */
status = DFANaddfid(file_id, file_label);

/* Write the description to the file. */
status = DFANaddfds(file_id, file_desc, strlen(file_desc));

/* Close the file. */
status = Hclose(file_id);

}
```

FORTRAN:

```
      PROGRAM CREATE ANNOTATION

       character*50 file_label, file_desc
       integer daafid, daafds, status, file_id, hopen, hclose

       integer*4 DFACC_CREATE
       parameter (DFACC_CREATE = 4)

       file_label = "This is a file label."
       file_desc = "This is a file description."

C      Open the HDF file to write the annotations.
       file_id = hopen('Example1.hdf', DFACC_CREATE, 0)

C      Write the label to the file.
       status = daafid(file_id, file_label)

C      Write the description to the file.
       status = daafds(file_id, file_desc, 26)

C      Close the file.
       status = hclose(file_id)

       end
```

10.5.3 Assigning an Object Label: DFANputlabel

To write a file label, the calling program must contain a call to **DFANputlabel**:

C: `status = DFANputlabel(filename, tag, ref, label);`

FORTRAN: `status = daplab(filename, tag, ref, label)`

DFANputlabel has four parameters: `filename`, `tag`, `ref`, and `label`. The `label` parameter contains a single null-terminated string that defines the annotation.

The parameters of **DFANputlabel** are further defined in Table 10F below.

10.5.4 Assigning an Object Description: DFANputdesc

To write an object description, the calling program must contain a call to **DFANputdesc**:

C:
```
status = DFANputdesc(filename, tag, ref, description, desc_len);
```

FORTRAN:
```
status = dapdesc(filename, tag, ref, description, desc_len)
```

DFANputdesc has five parameters: `filename`, `tag`, `ref`, `description`, and `desc_len`. The `filename` parameter is the name of the HDF file containing the object to be annotated. The `tag` and `ref` parameters are the tag/reference number pair of the object to be annotated. The `description` parameter contains a buffer for the annotation text and the `desc_len` parameter specifies the length of the buffer.

The parameters of **DFANputdesc** are further defined in the following table.

TABLE 10D **DFANputlabel and DFANputdesc Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANputlabel (daplab)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag of the object to be annotated.
	ref	uint16	integer	Reference number of the object to be annotated.
	label	char *	character* (*)	Object label string.
DFANputdesc (dapdesc)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag of the object to be annotated.
	ref	uint16	integer	Reference number of the object to be annotated.
	label	char *	character* (*)	Object description string.
	desc_len	int32	integer	Length of the description in bytes.

EXAMPLE 4. **Writing an Object Label and Description to a Scientific Data Set**

These examples illustrate the use of **DFANputlabel** and **DFANputdesc** to assign both an object label and an object description to a scientific data set immediately after it is written to file. The tag for scientific data sets is `DFTAG_NDG`.

C:
```
#include "hdf.h"

#define X_LENGTH 3
#define Y_LENGTH 2
#define Z_LENGTH 5

main( )
{

/* Create the data array. */
static float32 sds_data[X_LENGTH][Y_LENGTH][Z_LENGTH] =
{  1,  2,  3,  4,  5,
   6,  7,  8,  9, 10,
```

```
  11, 12, 13, 14, 15,
  16, 17, 18, 19, 20,
  21, 22, 23, 24, 25,
  26, 27, 28, 29, 30 };

/*
* Create the array that will hold the dimensions of
* the data array.
*/
int32 dims[3] = {X_LENGTH, Y_LENGTH, Z_LENGTH};
intn refnum, status;
static char object_desc[] = "This is an object description.";
static char object_label[] = "This is an object label.";

/* Write the data to the HDF file. */
status = DFSDadddata("Example1.hdf", 3, dims, (VOIDP)sds_data);

/* Get the reference number for the newly written data set. */
refnum = DFSDlastref( );

/* Assign the object label to the scientific data set. */
status = DFANputlabel("Example1.hdf", DFTAG_NDG, refnum, \
                object_label);

/* Assign the object description to the scientific data set. */
status = DFANputdesc("Example1.hdf", DFTAG_NDG, refnum, \
                object_desc, strlen(object_desc));

}
```

FORTRAN:

```
     PROGRAM ANNOTATE OBJECT

      integer dsadata, dims(3), status, refnum
      integer daplab, dapdesc, dslref

      integer*4 DFTAG_NDG, X_LENGTH, Y_LENGTH, Z_LENGTH
      parameter(DFTAG_NDG = 720,
     +          X_LENGTH = 5,
     +          Y_LENGTH = 2,
     +          Z_LENGTH = 3)

C     Create the data array.
      real*4 sds_data(X_LENGTH, Y_LENGTH, Z_LENGTH)
      data sds_data /
     +           1,  2,  3,  4,  5,
     +           6,  7,  8,  9, 10,
     +          11, 12, 13, 14, 15,
     +          16, 17, 18, 19, 20,
     +          21, 22, 23, 24, 25,
     +          26, 27, 28, 29, 30  /

C     Create the array the will hold the dimensions of the data array.
      data dims /X_LENGTH, Y_LENGTH, Z_LENGTH/

C     Write the data to the HDF file.
      ref = dsadata('Example1.hdf', 3, dims, sds_data)

C     Get the reference number for the newly written data set.
      refnum = dslref( )

C     Assign the object label to the scientific data set.
```

```
      status = daplab('Example1.hdf', DFTAG_NDG, refnum,
     +              'This is an object label.')

C     Assign an object description to the scientific data set.
      status = dapdesc('Example1.hdf', DFTAG_NDG, refnum,
     +               'This is an object description.', 30)

      end
```

10.6 Reading Annotations

The DFAN interface provides two functions for reading file annotations, which are described below.

10.6.1 Reading a File Label: DFANgetfidlen and DFANgetfid

The DFAN programming model for reading a file label is as follows:

1. Get the length of the label.
2. Read the file label.

To read the first file label in a file, the calling program must contain the following function calls:

C:

```
isfirst = 1;
label_length = DFANgetfidlen(file_id, isfirst);
label_buffer = HDgetspace(label_length);
fid_len = DFANgetfid(file_id, label_buffer, label_length,
                              isfirst);
```

FORTRAN:

```
isfirst = 1
label_length = dagfidl(file_id, isfirst)
fid_len = dagfid(file_id, label_buffer, label_length, isfirst)
```

DFANgetfidlen has two parameters: `file_id` and `isfirst`. The `isfirst` parameter specifies whether the first or subsequent file annotations are to be read. To read the first file label length `isfirst` should be set to the value `1`, to sequentially step through all the remaining file labels assigned to a file `isfirst` should be set to `0`.

When **DFANgetfidlen** is first called for a given file, it returns the length of the first file label. To get the lengths of subsequent file labels, you must call **DFANgetfid** between calls to **DFANgetfidlen**. Otherwise, additional calls to **DFANgetfidlen** will return the length of the same file label.

DFANgetfid has four parameters: `file_id`, `label_buffer`, `label_length`, and `isfirst`. The `label_buffer` parameter is a pointer to a buffer for the label text. The `label_length` parameter is the length of the buffer in memory which can be shorter than the full length of the label in the file. If the `label_length` is not large enough, the label is truncated to `label_length-1` characters in the buffer `label_buffer`. The `isfirst` parameter is used to read the first or subsequent file annotations. To read the first file label `isfirst` should be set to `1`, to sequentially step through all the remaining file labels assigned to a file `isfirst` should be set to `0`.

HDgetspace is described in Chapter 2, titled *HDF Fundamentals*.

The parameters of **DFANgetfidlen** and **DFANgetfid** are described below. (See Table 10E on page 256.)

10.6.2 Reading a File Description: DFANgetfdslen and DFANgetfds

The DFAN programming model for reading a file description is as follows:

1. Get the length of the description.
2. Read the file description.

To read the first file description in a file, the calling program must contain the following calls:

C:
```
isfirst = 1;
desc_length = DFANgetfdslen(file_id, isfirst);
desc_buffer = HDgetspace(desc_length);
fds_len = DFANgetfds(file_id, desc_buf, desc_length, isfirst);
```

FORTRAN:
```
isfirst = 1
desc_length = dagfdsl(file_id, isfirst)
fds_len = dagfds(file_id, desc_buf, desc_length, isfirst)
```

DFANgetfdslen has two parameters: `file_id` and `isfirst`. The `isfirst` parameter specifies whether the first or subsequent file annotations are to be read. To read the first file description length `isfirst` should be set to the value `1`, to sequentially step through all the remaining file descriptions assigned to a file `isfirst` should be set to `0`.

When **DFANgetfdslen** is first called for a given file, it returns the length of the first file description. To get the lengths of successive file descriptions, you must call **DFANgetfds** between calls to **DFANgetfdslen** as with **DFANfidgetlen**.

DFANgetfds has four parameters: `file_id`, `desc_buf`, `desc_length`, and `isfirst`. The `desc_buffer` parameter is a pointer to a buffer for the description text. The `desc_length` parameter is the length of the buffer in memory which can be shorter than the full length of the description in the file. If the `desc_length` is not large enough, the description is truncated to `desc_length` characters in the buffer `desc_buf`. The `isfirst` parameter specifies whether the first or subsequent file annotations are to be read. To read the first file description `isfirst` should be set to the value `1`, to sequentially step through all the remaining file descriptions assigned to a file `isfirst` should be set to `0`.

The parameters of these routines are described further in the following table.

TABLE 10E

DFANgetfidlen, DFANgetfid, DFANgetfdslen and DFANgetfds Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANgetfidlen (dagfidl)	file_id	int32	integer	File identifier.
	isfirst	intn	integer	Location of the next annotation.
DFANgetfid (dagfid)	file_id	int32	integer	File identifier.
	desc_buf	char *	character* (*)	File label buffer.
	buf_length	int32	integer	Label buffer length.
	isfirst	intn	integer	Location of the next annotation
DFANgetfdslen (dagfdsl)	file_id	int32	integer	File identifier.
	isfirst	intn	integer	Location of the next annotation.
DFANgetfds (dagfds)	file_id	int32	integer	File identifier.
	description	char *	character* (*)	File description buffer.
	desc_length	int32	integer	Description buffer length.
	isfirst	intn	integer	Location of the next annotation.

EXAMPLE 5.

Reading a File Label and a File Description

The following examples read a file label from the HDF file named "Example1.hdf". The **DFANgetfidlen** routine is used to verify the length of the label before the read operation is performed. The argument "1" in both routines indicate the first description in the HDF file is the target. **DFANgetfdslen** and **DFANgetfds** can be directly substituted for **DFANgetfidlen** and **DFANgetfid** in order to read a file description instead of a file label.

C:

```
#include "hdf.h"

main( )
{
int32 file_id, file_label_len;
char *file_label;
intn status;

/* Open the HDF file containing the annotation. */
file_id = Hopen("Example1.hdf", DFACC_READ, 0);

/* Determine the length of the file label. */
file_label_len = DFANgetfidlen(file_id, 1);

/* Allocated memory for the file label buffer. */
file_label = HDgetspace(file_label_len);

/* Read the file label. */
file_label_len = DFANgetfid(file_id, file_label, file_label_len, 1);

/* Close the file */
status = Hclose(file_id);

}
```

FORTRAN:

```
     PROGRAM GET ANNOTATION

      integer status, file_id, label_length
      integer hopen, hclose, dagfidl, dagfid
      character file_label(50)

      integer*4 DFACC_READ
      parameter(DFACC_READ = 1)

C     Open the HDF file containing the file label.
      file_id = hopen("Example1.hdf", DFACC_READ, 0)

C     Determine the length of the file label.
      label_length = dagfidl(file_id, 1)

C     Read the file label.
      status = dagfid(file_id, file_label, label_length, 1)

C     Close the HDF file.
      status = hclose(file_id)

      end
```

10.6.3 Reading an Object Label: DFANgetlablen and DFANgetlabel

The DFAN programming model for reading a object label is as follows:

1. Get the length of the label.
2. Read the file label.

To read the first object label in a file, the calling program must contain the following routines:

C:
```
label_length = DFANgetlablen(filename, tag, ref);
label_buf = HDgetspace(label_length);
status = DFANgetlabel(filename, tag, ref, label_buf,
                         label_length);
```

FORTRAN:
```
label_length = daglabl(filename, tag, ref)
status = daglab(filename, tag, ref, label_buf, label_length)
```

DFANgetlablen returns the length of the label assigned to the object identified by the given tag/reference number pair. **DFANgetlabel** must be called between calls to **DFANgetlablen** - it is the routine that actually returns the label and prepares the API to read the next label.

DFANgetlabel has five parameters: `filename`, `tag`, `ref`, `label_buf`, and `label_length`. The `label_buf` parameter is a pointer to a buffer that stores the label text. The `label_length` parameter is the length of the buffer in memory - this can be shorter than the full length of the label in the file. If the `label_length` is not large enough, the label is truncated to `label_length` characters in the buffer `label_buf`. The length of `label_buf` must be at least one greater than the anticipated length of the label, to account for the null termination appended to the label text.

The parameters of **DFANgetlablen** and **DFANgetlabel** are defined below.

10.6.4 Reading an Object Description: DFANgetdesclen and DFANgetdesc

The DFAN programming model for reading a object description is as follows:

1. Get the length of the description.
2. Read the file description.

To read the first object description in a file, the calling program must contain the following routines:

C:
```
desc_length = DFANgetdesclen(filename, tag, ref);
desc_buf = HDgetspace(desc_length);
status = DFANgetdesc(filename, tag, ref, desc_buf, desc_length);
```

FORTRAN:
```
label_length = dagdlen(filename, tag, ref)
status = dagdesc(filename, tag, ref, desc_buf, desc_length)
```

DFANgetdesclen returns the length of the description assigned to the object identified by the specified tag/reference number pair. **DFANgetdesc** must be called between calls to **DFANgetdesclen** to reset the current object description to the next in the file.

DFANgetdesc takes five parameters: `filename`, `tag`, `ref`, `desc_buf`, and `desc_length`. The `desc_buf` parameter is a pointer to the buffer that stores the description text. The `desc_length` parameter is the length of the buffer in memory which can be shorter than the full length of the description in the file. If the `desc_length` is not large enough, the description is truncated to `desc_length` characters in the buffer `desc_buf`.

The parameters of **DFANgetdesclen** and **DFANgetdesc** are defined in the following table.

TABLE 10F

DFANgetlablen, DFANgetlabel, DFANgetdesc and DFANgetdesclen Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANgetlablen (dagllen)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag assigned to the annotated object.
	ref	uint16	integer	Reference number for the annotated object.
DFANgetlabel (daglab)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag assigned to the annotated object.
	ref	uint16	integer	Reference number assigned to the annotated object.
	label_buf	char *	character* (*)	Buffer for the returned annotation.
	label_length	int32	integer	Size of the buffer allocated to hold the annotation.
DFANgetdesclen (dagdlen)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag assigned to the annotated object.
	ref	uint16	integer	Reference number for the annotated object.
DFANgetdesc (dagdesc)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag assigned to the annotated object.
	ref	uint16	integer	Reference number assigned to the annotated object.
	desc_buf	char *	character* (*)	Buffer for the returned annotation.
	desc_length	int32	integer	Size of the buffer allocated to hold the annotation.

EXAMPLE 6.

Reading an Object Label and Description

The following examples demonstrate the use of **DFANgetdesclen** and **DFANgetdesc** to read an object description assigned to a scientific data set. These examples assume that, in addition to other data objects, the "Example1.hdf" HDF file also contains multiple scientific data sets, some of which may not be annotated. **Hfind** is used to determine the reference number for the first annotated scientific data object in the file.

C:

```
#include "hdf.h"

main( )
{
intn desc_length = -1, status;
char desc[50];
int32 file_id;
uint16 tag = 0, ref = 0;
uint32 find_offset, find_length;

/* Open the file and initialize the searching parameters to 0. */
file_id = Hopen("Example1.hdf", DFACC_READ, 0);

/*
* Start a sequential forward search for the first reference
* number assigned to a scientific data set.
*/
while (Hfind(file_id, DFTAG_NDG, DFREF_WILDCARD, &tag, &ref, \
    &find_offset, &find_length, DF_FORWARD) != FAIL) {

/*
```

```
* After discovering a valid reference number, check for an
* object description by returning the length of the description.
* If the inquiry fails, continue searching for the next valid
* reference number assigned to a scientific data set.
*/
if ((desc_length = DFANgetdesclen("Example1.hdf", tag, ref)) \
    == FAIL)
    break;

/*
* If a description exists and it will fit in the description buffer,
* print it.
*/
if (desc_length != FAIL && desc_length <= 50) {
    status = DFANgetdesc("Example1.hdf", tag, ref, desc, desc_length);
    printf("Description: %s\n", desc);
}
}

/* Close the file. */
status = Hclose(file_id);

}
```

FORTRAN:

There is no Fortran-77 version of the Example 4 C code for this version of the documentation as there is no Fortran-77 equivalent of **Hfind**.

10.7 Maintenance Routines

The DFAN interface provides only function for interface maintenance - **DFANclear**.

10.7.1 Clearing the DFAN Interface's Internal Structures and Settings: DFANclear

DFANclear clears all internal library structures and parameters of the DFAN annotiation interface.

DFANclear returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-1`) otherwise.

When a file is regenerated in a single run by a library routine of another interface (such as **DFSDputdata**), **DFANclear** should be called to reset the interface

The parameters are described further in the following table.

TABLE 10G **DFANclear Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANclear (daclear)	None	None	None	None.

10.8 Determining Reference Numbers

It is advisable to check the reference number before attempting to assign an object annotation, as the overwriting of reference numbers isn't prevented by the HDF library routines.

There are three ways to check a reference number for an object:

- Access the object with a read or write operation followed by **DF*lastref**.
- Call **DFANlablist** to return a list of all assigned reference numbers for a given tag.
- Call **Hfind** to locate an object with a given tag/reference number pair.

10.8.1 Determine a Reference Number for the Last Object Accessed: DF*lastref

There are two methods of obtaining a reference number through the use of a **DF*lastref** call. The first approach is to obtain and store the reference number of an object immediately after the object is created:

1. Create the data object.
2. Call **DF*lastref** to determine its reference number.
3. Read or write an object annotation.

The second approach is to determine the reference number at some time after the data object is created. This approach requires repeated **DF*read** calls until the appropriate object is accessed, followed by a call to **DF*lastref**:

1. Read the appropriate data object.
2. Call **DF*lastref** to determine its reference number.
3. Read or write and object annotation.

Most HDF interfaces provide one routine that assigns a specified reference number to a data object and another routine that returns the reference number for the last data object accessed. (See Table 10H.) However, the SD interface doesn't. Also, the DFAN annotation doesn't include a **DF*lastref** routine.

Although **DF*writeref** calls are designed to assign specific reference numbers, they are not recommended for general use because there is no protection against reassigning an existing reference number and overwriting data. In general, it is better to determine a reference number for a data object by calling **DF*lastref** immediately after reading or writing a data object.

The **DF*lastref** routines have no parameters. The **DF*writeref** routines have two: `filename`, which is the name of the file that contains the data object, and `ref`, which is the reference number for the next data object read operation.

TABLE 10H **List and Descriptions of the DF*writeref and DF*lastref Routines**

HDF Data Object	Routine Name (Fortran-77)	Description
8-bit Raster Image	DFR8writeref (d8wref)	Assigns the specified number as the reference number for the next 8-bit raster write operation and updates the write counter to the reflect highest reference number.
	DFR8lastref (d8lref)	Returns the reference number for the last 8-bit raster image set accessed.

HDF Data Object	Routine Name (Fortran-77)	Description
24-bit Raster Image	**DF24writeref (d2wref)**	Assigns the specified number as the reference number for the next 24-bit raster write operation and updates the write counter to reflect the highest reference number.
	DF24lastref (d2lref)	Returns the reference number for the last 24-bit raster image set accessed.
Palette	**DFPwriteref (dpwref)**	Assigns the specified number as the reference number for the next palette write operation and updates the write counter to reflect the highest reference number.
	DFPlastref (dplref)	Returns the reference number for the last palette accessed.
DFSD Scientific Data	**DFSDwriteref (dswref)**	Assigns the specified number as the reference number for the next SDS write operation and updates the write counter to reflect the highest reference number.
	DFSDlastref (dslref)	Returns the reference number for the last scientific data set accessed.
Annotation	**DFANlastref (dalref)**	Returns the reference number for the last annotation accessed.

10.8.2 Querying a List of Reference Numbers for a Given Tag: DFANlablist

Given a tag and two buffers, **DFANlablist** will fill one buffer with all reference numbers for the given tag and the other with all labels assigned to the given tag. The programming model for determining a list of reference numbers is as follows:

1. Determine the number of reference numbers that exist for a given tag.
2. Allocate a buffer to store the reference numbers.
3. Specify the maximum label length.
4. Allocate a buffer to store the labels.
5. Store the list of reference numbers and their labels.

To create a list of reference numbers and their labels for a given tag, the following routines should be called:

C:

```
num_refs = Hnumber(file_id, tag);
ref_buf = HDmalloc(sizeof(uint16*)*num_refs);
max_lab_len = 16;
label_buf = HDmalloc(max_lab_len * num_refs);
start_pos = 0;
num_of_refs = DFANlablist(filename, tag, ref_buf, label_buf,
                                num_refs, max_lab_len,
                                start_pos);
```

FORTRAN:

```
num_refs = hnumber(file_id, tag)
max_lab_len = 16
start_pos = 0
num_of_refs = dallist(filename, tag, ref_buf, label_buf,
                            num_refs, max_lab_len, start_pos)
```

Hnumber determines how many objects with the specified tag are in a file. It is described in Chapter 2, titled *HDF Fundamentals*.

DFANlablist has seven parameters: `filename`, `tag`, `ref_list`, `label_buf`, `num_refs`, `max_lab_len`, and `start_pos`. The `filename` parameter specifies the name of the file to search and `tag` specifies the search tag to use when creating the reference and label list. The `ref_buf` and `label_buf` parameters are buffers used to store the reference numbers and labels associated with `tag`. The `num_ref` parameter specifies the length of the reference number list and the max_lab_len parameter specifies the maximum length of a label. The start_pos parameter specifies the first label to read. For instance, if start_pos has a value of 1 all labels will be read - if it has a value of 4, all but the first three labels will be read.

Taken together, the ref_list and label_list constitute a directory of all objects and their labels for a given tag. The contents of label_list can be displayed to show all of the labels for a given tag or it can be searched to find the reference number of a data object with a certain label. Once the reference number for a given label is found, the corresponding data object can be accessed by invoking other HDF routines. Therefore, this routine provides a mechanism for direct access to data objects in HDF files.

TABLE 10I

DFANlablist Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
DFANlablist (dallist)	filename	char *	character* (*)	Name of the file to be accessed.
	tag	uint16	integer	Tag assigned to the annotated object.
	ref_list	uint16 []	integer (*)	Reference number for the annotated object.
	label_list	char *	character* (*)	Buffer for the labels.
	list_len	int	integer	Size of the reference number and label lists.
	label_len	intn	integer	Maximum label length.
	start_pos	intn	integer	First entry in the reference number and label lists to be returned.

EXAMPLE 7.

Getting a List of Labels for All Scientific Data Sets

These examples illustrate the method used to get a list of all labels used in scientific data sets in an HDF file using **DFANlablist.** The DFS_MAXLEN definition is located in the "hlimits.h" include file.

C:

```
#include "hdf.h"

#define LISTSIZE 20

main( )
{

int i, num_of_labels, start_position = 1, list_length = 10;
uint16 ref_list[LISTSIZE];
char label_list[DFS_MAXLEN*LISTSIZE-1];

/* Get the total number of labels in the "Example1.hdf" file. */
num_of_labels = DFANlablist("Example1.hdf", DFTAG_NDG, ref_list, \
          label_list, list_length, DFS_MAXLEN,   \
          start_position);

/*
* Print the reference numbers and label names for each label
* in the list.
*/
for (i = 0; i < num_of_labels; i++)
  printf("\n\t%d\tRef number: %d\tLabel: %s", i+1, ref_list[i], \
         label_list - (i * 13));
```

```
printf("\n");

}
```

FORTRAN:

```
      PROGRAM GET LABEL LIST

       integer dallist
       integer*4 DFTAG_NDG, LISTSIZE, DFS_MAXLEN

       parameter (DFTAG_NDG = 720,
      +          LISTSIZE = 20,
      +          DFS_MAXLEN = 255)

       character*60 label_list(DFS_MAXLEN*LISTSIZE)
       integer i, num_of_labels, start_position, ref_list(DFS_MAXLEN)

       start_position = 1

       num_of_labels = dallist('Example1.hdf', DFTAG_NDG, ref_list,
      +                        label_list, 10, DFS_MAXLEN,
      +                        start_position)

       do 10 i = 1, num_of_labels
         print *,'    Ref number:  ',ref_list(i),
      +          '    Label: ',label_list(i)
10     continue

       end
```

10.8.3 Locate an Object by Its Tag and Reference Number: Hfind

Instead of using **DFANlablist** to create a list of reference numbers to search, HDF provides a general search routine called **Hfind**. **Hfind** is described in Chapter 2, titled *HDF Fundamentals*.

10.9 The Multifile Annotation Interface

The C routine names of the multifile annotation interface are prefaced by the string "AN" and the Fortran-77 routine names are prefaced by "af". The purpose of the AN multifile annotation interface is to permit concurrent operations on a set of annotations that exist in more than one file rather than requiring the program to deal with the annotations on a file-by-file basis. The design of the AN interface is similar to the multifile interfaces for raster image (GR) and scientific data set objects (SD).

10.9.1 AN Library Routines

These functions are divided into the following categories:

- ***Access routines*** initialize the interface and provide and terminate access to annotations.
- ***Read/write functions*** read and write file or object annotations.
- ***General inquiry functions*** return information about the annotations in a file.

The AN function calls are defined in the following table and on their respective pages in the *HDF Reference Guide*.

TABLE 10J **AN Library Routines**

Category	Routine Names C	Routine Names Fortran-77	Description
Access	ANstart	afstart	Initialize AN interface and open file.
	ANselect	afselect	Gets an annotation identifier from a specified index.
	ANend	afend	Close file and AN interface.
	ANcreate	afcreate	Create a new data annotation.
	ANcreatef	affcreate	Create a new file annotation.
	ANendaccess	afendaccess	Terminate access to the annotation that corresponds to the given annotation identifier.
Read/write	ANwriteann	afwriteann	Write the annotation the corresponds to the given annotation identifier.
	ANreadann	afreadeann	Read the annotation the corresponds to the given annotation identifier.
General Inquiry	ANnumann	afnumann	Get the number of annotations in the file that correspond to a given type and tag/reference number pair.
	ANannlist	afannlist	Get a list of annotations in the file that correspond to a given type and tag/reference number pair.
	ANannlen	anannlen	Get the length of the annotation corresponding to a given annotation identifier.
	ANfileinfo	anfileinfo	Get information concerning the number of annotations of each type in the file.
	ANget_tagref	afgettagref	Get a tag/reference number pair from a specified annotation type and index.
	ANid2tagref	afidtagref	Get a tag/reference number pair from a specified annotation id.
	ANtagref2id	aftagrefid	Get an annotation id from a specified tag/reference number pair.
	ANatype2tag	afatypetag	Get an object tag corresponding to a specified annotation type.
	ANtag2atype	aftagatype	Get an annotation type corresponding to a specified object tag.

10.9.2 Type Definitions Used in the Multifile Annotation Interface

The multifile annotation interface uses the four general annotation types used in HDF: the data label, the data description, the file label and the file description. These annotation types correspondingly map to the `AN_DATA_LABEL`, the `AN_DATA_DESC`, the `AN_FILE_LABEL` and the `AN_FILE_DESC` definitions. Several routines in the multifile annotation interface require one of these type definitions to be passed in as an argument to designate the kind of annotation to be created or accessed.

10.10 Programming Model for the Multifile Annotation Interface

The programming model for the multifile annotation interface is as follows:

1. Open a file.
2. Initialize the multifile annotation interface and obtain an interface id.
3. Create or obtain an annotation id.
4. Perform the desired annotation operation.
5. Terminate access to the annotation interface and the annotation.
6. Close the file.

As with the GR and SD interfaces an identifier to the object to be created or accessed is obtained when the interface is initialized. This programming model allows several files to be opened and the contents operated on as long as the calling program accurately keeps track of each file, interface and object identifier returned by the interface. Each object and file must be explicitly closed before the termination of the calling program.

10.11 Creating and Writing Annotations Using the AN Interface

The AN interface writes file labels, file descriptions, data object labels and data object descriptions according to the general AN programming models.

10.11.1 Creating Annotations

Creating an annotation without writing it involves the following steps:

1. Open the HDF file.
2. Initialize the AN interface.
3. Define the type of annotation to be created.
4. Create the annotation.
5. Terminate access to the annotation.
6. Terminate access to the AN interface.
7. Close the file.

To create a file annotation, the calling program must contain the following AN routine calls:

C:

```
file_id = Hopen(filename, access. block_size);
an_id = ANstart(file_id);
ann_id = ANcreatef(an_id, ann_type);
status = ANendaccess(ann_id);
status = ANend(an_id);
status = Hclose(file_id);
```

FORTRAN:

```
file_id = hopen(filename, access, block_size)
an_id = afstart(file_id)
ann_id = affcreate(an_id, ann_type)
status = afendaccess(ann_id)
status = afend(an_id)
status = hclose(file_id)
```

ANstart initializes the multifile annotation interface**. ANcreate** and **ANcreatef** creates the annotation**. ANendaccess** terminates access to the annotation, and **ANend** terminates access to the multifile annotation interface**.**

ANcreate has four parameters: `an_id`, `tag`, `ref` and `ann_type`. The `an_id` parameter is the interface identifier for the annotation to be written, the `tag` and `ref` parameters are the tag/reference number pair of the object the annotation will be assigned to and the `ann_type` parameter is the data annotation type. It is set to either `AN_DATA_LABEL` or `AN_DATA_DESC`. These annotation type definitions are defined in the "hdf.h" header file.

ANcreatef is used to create file annotations. The `ann_type` parameter in a call to **ANcreatef** can only be set to the type definitions `AN_FILE_LABEL` or `AN_FILE_DESC`, which are also defined in the "hdf.h" header file.

ANendaccess's one argument, `ann_id`, is an annotation returned by **ANcreate** and is disposed of by **ANendaccess**. Any subsequent attempts to access the annotation id will result in an error condition.

ANend also has only one parameter, `an_id`, which is the AN interface identifier returned by **ANstart.** This identifier is disposed of by **ANend** and access to the interface routines is terminated during this call. Any subsequent attempts to access the annotation interface id or routines will result in an error condition.

The parameters of these functions are defined in the following table.

TABLE 10K **ANstart, ANcreate, ANcreatef, ANendaccess and ANend Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANstart (afstart)	file_id	int32	integer	File identifier.
ANcreate (afcreate)	an_id	int32	integer	AN interface identifier.
	tag	uint16	integer	Tag of the item to be assigned the data annotation.
	ref	uint16	integer	Reference number of the item to be assigned the data annotation.
	type	int32	integer	Data annotation type: label or description.
ANcreatef (affcreate)	an_id	int32	integer	AN interface identifier.
	type	int32	integer	File annotation type: label or description.
ANendaccess (afendaccess)	ann_id	int32	integer	Annotation identifier.
ANend (afend)	an_id	int32	integer	AN interface identifier.

10.11.2 Writing an Annotation: ANwriteann

The AN programming model for writing an annotation is as follows:

1. Open the HDF file.
2. Initialize the AN interface.
3. Create the annotation identifier.
4. Write the annotation.
5. Terminate access to the annotation.
6. Terminate access to the AN interface.
7. Close the HDF file.

To write a file label, the calling program must contain the following AN routine calls:

C:
```
file_id = Hopen(filename, access. block_size);
an_id = ANstart(file_id);
ann_id = ANcreatef(an_id, ann_type);
status = ANwriteann(ann_id, label, HDstrlen(label));
status = ANendaccess(ann_id);
status = ANend(an_id);
status = Hclose(file_id);
```

FORTRAN:
```
file_id = hopen(filename, access, block_size)
an_id = afstart(file_id)
ann_id = affcreate(an_id, ann_type)
```

```
status = afwriteann(ann_id, label, ann_length)
status = anendaccess(ann_id)
status = afend(an_id)
status = hclose(file_id)
```

ANwriteann has three parameters: `ann_id`, `label` and `ann_length`. The `ann_id` parameter is the identifier for the annotation to be written, the `label` parameter contains the annotation string and the `ann_length` parameter contains the length of the annotation string. The label must be a null-terminated string. In the C interface, the **HDstrlen** function can be used to dynamically determine the length of the label string. In either C or Fortran-77, specifying a length value different from the actual length of the label will result in the label either being truncated or null-padded accordingly.

The parameters of **ANwriteann** are further defined in the following table.

TABLE 10L

ANwriteann Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANwriteann (afwriteann)	ann_id	int32	integer	Identifier for the annotation to be written.
	label	char *	character* (*)	Label of the annotation.
	ann_length	int32	integer	Length of the label in bytes.

EXAMPLE 8.

Adding a File Label Using the AN Interface

The following examples add a file label and description to the HDF file named "Example6.hdf" using the AN interface routines.

C:

```
#include "hdf.h"
#include <string.h>

main( )
{

int32 file_id, an_id, ann_id, vgroup_id, ann_type;
uint16 obj_tag, obj_ref;
intn status;
static char file_label[] = "This is a file label.";

/* Create the HDF file. */
file_id = Hopen("Example6.hdf", DFACC_CREATE, 0);

/* Initialize the AN interface and obtain an interface id. */
an_id = ANstart(file_id);

/* Set the file annotation type to be a file label. */
ann_type = AN_FILE_LABEL;

/* Create the file label and obtain an annotation id. */
ann_id = ANcreatef(an_id, ann_type);

/* Write the label to the file. */
status = ANwriteann(ann_id, file_label, strlen(file_label));

/* Create a vgroup. */
status = Vstart(file_id);
vgroup_id = Vattach(file_id, -1, "w");
```

```
status = Vdetach(vgroup_id);
status = Vend(file_id);

obj_tag = DFTAG_VG;
obj_ref = 0;

/* Create the data description label. */
ann_id = ANcreate(an_id, obj_tag, obj_ref, AN_DATA_DESC);

/* Write the label to the file. */
status = ANwriteann(ann_id, file_label, strlen(file_label));

/* Terminate access to the annotation. */
status = ANendaccess(ann_id);

/* Terminate access to the AN interface. */
status = ANend(an_id);

/* Close the file. */
status = Hclose(file_id);

}
```

FORTRAN:

```
      PROGRAM CREATE A FILE LABEL

       integer hopen, afstart, affcreate, afendaccess, afend
       integer hclose, status, file_id, an_id, ann_id
       integer afwriteann

       integer*4 DFACC_CREATE, AN_FILE_LABEL
       parameter (DFACC_CREATE = 4, AN_FILE_LABEL = 2)

       character*21 file_label /"This is a file label."/

C      Open the HDF file.
       file_id = hopen('Example6.hdf', DFACC_CREATE, 0)

C      Initialize the AN interface and obtain an interface id.
       an_id = afstart(file_id)

C      Use the file_id to write the label to the file.
       ann_id = affcreate(an_id, AN_FILE_LABEL)

C      Write the label to the file.
       status = afwriteann(ann_id, file_label, 21)

C      Terminate access to the annotation.
       status = afendaccess(ann_id)

C      Terminate access to the AN interface.
       status = afend(an_id)

C      Close the file.
       status = hclose(file_id)

       end
```

10.12 Reading Annotations Using the AN Interface

As is the case with writing annotations there is only one AN routine used for reading annotations. This routine is described below.

10.12.1 Reading an Annotation: ANreadann

The AN programming model for reading an annotation is similar to the model for writing one:

1. Open the HDF file.
2. Initialize the AN interface.
3. Read the annotation.
4. Terminate access to the annotation.
5. Terminate access to the AN interface.
6. Close the HDF file.

To read a file label, the calling program must contain the following AN routine calls:

C:

```
file_id = Hopen(filename, access. block_size);
an_id = ANstart(file_id);
status = ANreadann(ann_id, buf, HDstrlen(buf));
status = ANendaccess(ann_id);
status = ANend(an_id);
status = Hclose(file_id);
```

FORTRAN:

```
file_id = hopen(filename, access, block_size)
an_id = afstart(file_id)
status = afreadann(ann_id, buf, buf_length)
status = anendaccess(ann_id)
status = afend(an_id)
status = hclose(file_id)
```

ANreadann has three parameters: `ann_id`, `buf` and `buf_length`. The `ann_id` parameter is the identifier for the annotation to be read, the `buf` parameter is the buffer the annotation string will be read into and the `buf_length` parameter is the length of the buffer. As with **ANwriteann**, the **HDstrlen** function can be used to dynamically determine the length of the buffer. In either C or Fortran-77, specifying a buffer length value different from the actual length of the buffer will result in the annotation either being truncated or null-padded accordingly.

The parameters of the **ANreadann** routine is further defined in the following table.

TABLE 10M **ANreadann Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANreadann (afreadann)	ann_id	int32	integer	Identifier of the annotation to be read.
	buf	char *	character* (*)	Buffer for the returned annotation.
	buf_length	int32	integer	Size of the buffer in bytes.

10.13 Obtaining Annotation Information Using the AN Interface

In some cases the HDF programmer must get annotation information for the purpose of either locating a particular annotation or set of annotations that correspond to a set of search criteria. The following sections describe the AN routines designed to do this.

10.13.1 Selecting an Annotation: ANselect

The **ANselect** function returns the `ann_id` corresponding to the index passed into the function as the second index parameter and the annotation type passed in as the third `annot_type` parameter. The interface identifier returned by **ANstart** must also be provided. The `annot_type` parameter must be one of the annotation type definitions listed in Section 10.9.2 on page 265. Upon unsuccessful completion **ANselect** returns a `FAIL` status code.

TABLE 10N **ANselect Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANselect (afselect)	an_id	int32	integer	AN interface identifier.
	index	int32	integer	Index of the target annotation.
	annot_type	int32	integer	Type of the target annotation.

10.13.2 Obtaining General Annotation Information: ANfileinfo

ANfileinfo returns the total number of file labels, file descriptors, data labels and data descriptors in the current file. The only input parameter required by **ANfileinfo** is the `an_id` passed as its first argument. The information provided by **ANfileinfo** can also be used as object indices. As such they can be passed into a call to **ANselect** within a loop for the purpose of accessing all annotations in a file.

The **ANfileinfo** routine returns either a `SUCCEED` or `FAIL` return code. These status codes are defined in the "hdf.h" header file.

TABLE 10O **ANfileinfo Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANfileinfo (affileinfo)	an_id	int32	integer	AN interface identifier.
	n_file_label	int32 *	integer (*)	Number of file labels in the file.
	n_file_desc	int32 *	integer (*)	Number of file descriptors in the file.
	n_data_label	int32 *	integer (*)	Number of data labels in the file.
	n_data_desc	int32 *	integer (*)	Number of data descriptors in the file.

10.13.3 Getting the Length of an Annotation: ANannlen

The **ANannlen** function returns either the length of the annotation corresponding to the `ann_id` passed in the function as it’s only argument or a `FAIL` return code upon unsuccessful completion.

TABLE 10P

ANannlen Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANannlen (afannlen)	ann_id	int32	integer	Identifier of the target annotation.

EXAMPLE 9.

Reading File Labels

The following examples reads the file label annotation in the HDF file named "Example6.hdf".

C:

```
#include "hdf.h"

main( )
{

int32 file_id, an_id, ann_id, ann_idx, i, ann_length;
int32 n_file_label, n_file_desc, n_data_label, n_data_desc;
int32 status;
char *ann_buf;

/* Open the HDF file for reading. */
file_id = Hopen("Example6.hdf", DFACC_RDWR, 0);

/* Initialize the AN interface and obtain an interface id. */
an_id = ANstart(file_id);

/* Get the annotation information. */
status = ANfileinfo(an_id, &n_file_label, &n_file_desc, \
                &n_data_label, &n_data_desc);

/* Get the file labels. */
for (i = 0; i < n_file_label; i++) {

    /* Get the identifier for the current file label. */
    ann_id = ANselect(an_id, i, AN_FILE_LABEL);

    /* Get the length of the file label. */
    ann_length = ANannlen(ann_id);

    /* Get the length of the file label. */
    ann_buf = HDmalloc(ann_length * sizeof(char));

    /* Read the file label. */
    status = ANreadann(ann_id, ann_buf, ann_length);
}

/* Terminate access to the annotation. */
status = ANendaccess(ann_id);

/* Terminate access to the AN interface. */
status = ANend(an_id);

/* Close the file. */
status = Hclose(file_id);

}
```

FORTRAN:

```
      PROGRAM GET LABELS

       integer*4 afstart, affileinfo, afselect, afendaccess, afend
       integer*4 afannlen, afendaccess, afend, hopen, hclose
       integer*4 status, file_id, an_id, ann_id, i
       character ann_buf(50)

       integer*4 DFACC_RDWR, AN_FILE_LABEL
       parameter (DFACC_RDWR = 3, AN_FILE_LABEL = 2)

C      Open the HDF file.
       file_id = hopen('Example6.hdf', DFACC_RDWR, 0)

C      Initialize the AN interface and obtain an interface id.
       an_id = afstart(file_id)

C      Get the annotation information.
       status = affileinfo(an_id, n_file_label, n_file_desc,
      +                   n_data_label, n_data_desc)

C      Get file label lengths.
       do 10 i = 0, n_file_label

C        Get the identifier for the current index.
         ann_id = afselect(i, AN_FILE_LABEL)

C        Get the length of the annotation for the current id.
         ann_length = afannlen(ann_id)

C        Read the file label.
         status = afreadann(ann_id, ann_buf, ann_length)

10     continue

C      Terminate access to the annotation.
       status = afendaccess(ann_id)

C      Terminate access to the AN interface.
       status = afend(an_id)

C      Close the file.
       status = hclose(file_id)

       end
```

10.13.4 Obtaining the Number of Annotations Corresponding to Given Search Criteria: ANnumann

The **ANnumann** routine returns the total number of annotations in the file that satisfy the search criteria supplied in the latter three parameters of the call. These parameters are: `type`, which is set to one of the type definitions listed in Section 10.9.2 on page 265, `obj_tag`, which is the tag of the object the annotation to be searched for is attached to, and `obj_ref`, which is the reference number of the object the annotation to be searched for is attached to. Upon unsuccessful completion **ANnumann** returns a `FAIL` status code.

TABLE 10Q

ANnumann Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANnumann (afnumann)	an_id	int32	integer	AN interface identifier.
	annot_type	int32	integer	Type of the target annotation.
	obj_tag	uint16	integer	Tag of the object the target annotation is attached to.
	obj_ref	uint16	integer	Reference number of the object the target annotation is attached to.

10.13.5 Obtaining the List of Annotations Corresponding to Given Search Criteria: ANannlist

The **ANannlist** routine is the counterpart to **ANnumann** in that, instead of returning the *number* of annotations that satisfy a given set of search criteria supplied in the latter three parameters of the call, it returns the *list* of annotation identifers in the file that satisfy the search criteria. **ANannlist**'s second, third and fourth parameters are the same as **ANnumann**'s. **ANannlist** returns the list in it's fifth parameter. Upon unsuccessful completion **ANannlist** also returns a `FAIL` status code.

TABLE 10R

ANannlist Parameter List

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANannlist (afannlist)	an_id	int32	integer	AN interface identifier.
	annot_type	int32	integer	Type of the target annotation.
	obj_tag	uint16	integer	Tag of the object the target annotation is attached to.
	obj_ref	uint16	integer	Reference number of the object the target annotation is attached to.
	ann_list	int32 *	integer (*)	Buffer for returned annotation identifiers that match the search criteria.

EXAMPLE 10.

Returning the Number of and the List of Object Descriptions

The following examples return the total number and list of object descriptions in the "Example6.hdf" file. The `DFTAG_DIA` tag is defined in the "htags.h" header file as a object description tag. The `AN_DATA_DESC` definition is in the "mfan.h" header file.

C:

```
#include "hdf.h"

main( )
{
int32 status, file_id, an_id, ann_id, ann_num;
int32 *ann_list;
uint16 obj_tag, obj_ref;
int32 annot_type;

/* Create the HDF file. */
file_id = Hopen("Example6.hdf", DFACC_RDWR, 0);

/* Initialize the AN interface and obtain an interface id. */
```

```
an_id = ANstart(file_id);

/* Set the annotation type to be a data description. */
annot_type = AN_DATA_DESC;

/* Set the tag to be a data identifier. */
obj_tag = DFTAG_DIA;

/* Set the reference number to be the first object. */
obj_ref = 0;

/* Get the number of object descriptions. */
ann_num = ANnumann(an_id, annot_type, obj_tag, obj_ref);

/* Allocate space for the annotation identifier. */
ann_list = HDmalloc(ann_num * sizeof(int32));

/* Get the list of object descriptions. */
status = ANannlist(an_id, annot_type, obj_tag, obj_ref, ann_list);

/* Terminate access to the annotation. */
status = ANendaccess(ann_id);

/* Terminate access to the AN interface. */
status = ANend(an_id);

/* Free the space allocated for the annotation identifier. */
HDfree(ann_list);

/* Close the file. */
status = Hclose(file_id);

}
```

FORTRAN:

```
      PROGRAM GET DATA DESC LIST

       integer hopen, afstart, affcreate, afendaccess, afend
       integer hclose status, file_id, an_id, ann_id
       integer annot_type, obj_tag, ann_num
       integer ann_list(50)

       integer*4 DFACC_CREATE, AN_DATA_DESC, DFTAG_DIA
       parameter (DFACC_CREATE = 4, AN_DATA_DESC = 1,
      +           DFTAG_DIA = 105)

C      Open the HDF file.
       file_id = hopen('Example6.hdf', DFACC_RDWR, 0)

C      Initialize the AN interface and obtain an interface id.
       an_id = afstart(file_id)

C      Set the annotation type to be a data description.
       annot_type = AN_DATA_DESC

C      Set the tag to be a data identifier description.
       obj_tag = DFTAG_DIA

C      Set the reference number to be the first object.
       obj_ref = 0

C      Get the number of file labels.
```

```
      ann_num = afnumann(an_id, annot_type, obj_tag, obj_ref)

C     Get the list of file labels.
      status = afannlist(an_id, annot_type, obj_tag, obj_ref, ann_list)

C     Terminate access to the annotation.
      status = afendaccess(ann_id)

C     Terminate access to the AN interface.
      status = afend(an_id)

C     Close the file.
      status = hclose(file_id)

      end
```

10.13.6 Obtaining the Tag/Reference Number Pair From a Specified Annotation Identifiers: ANget_tagref

The **ANget_tagref** routine returns the tag/reference number pair of the annotation identified by the specfied annotation identifier, index and annotation type.

The *ann_index* parameter is zero-based. The *ann_type* parameter value can be either `AN_DATA_LABEL`, `AN_FILE_LABEL`, `AN_DATA_DESC` or `AN_FILE_DESC`.

TABLE 10S **ANget_tagref Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANget_tagref (afgettagref)	an_id	int32	integer	AN interface identifier.
	ann_index	int32 *	integer	Index of the target annotation.
	ann_type	int32 *	integer	Annotation type of the target annotation.
	ann_tag	uint16 *	integer	Tag of the target annotation.
	ann_ref	uint16 *	integer	Reference number of the target annotation.

10.13.7 Obtaining the Tag/Reference Number Pair From a Specified Annotation Identifier: ANid2tagref

The **ANid2tagref** routine returns the tag/reference number pair of the annotation that is also identified by the specified annotation id.

ANid2tagref returns the tag/reference number pair if successful and `FAIL` (or `-1`) otherwise.

TABLE 10T **ANid2tagref Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANid2tagref (afidtagref)	an_id	int32	integer	AN interface identifier.
	ann_tag	uint16 *	integer	Tag of the target annotation.
	ann_ref	uint16 *	integer	Reference number of the target annotation.

10.13.8 Obtaining the Annotation Identifier From a Specified Tag/ Reference Number Pair: ANtagref2id

The **ANtagref2id** routine returns the identifier of the annotation that is also identified by the specified tag/reference number pair.

ANtagref2id returns the annotation identifier if successful and `FAIL` (or `-1`) otherwise.

TABLE 10U **ANtagref2id Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANtagref2id (aftagrefid)	an_id	int32	integer	AN interface identifier.
	ann_tag	uint16 *	integer	Type of the target annotation.
	ann_ref	uint16	integer	Tag of the object the target annotation is attached to.

10.13.9 Obtaining an Object Tag From a Specified Annotation Type: ANatype2tag

The **ANatype2tag** routine returns the object tag (prefaced by `DFTAG_`) corresponding to a given annotation type (prefaced by `AN_`).

Return values are `DFTAG_FID`, `DFTAG_FD`, `DFTAG_DIL or DFTAG_DIA`.

TABLE 10V **ANatype2tag Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANatype2tag	tag	int32 *	integer	Object type.

10.13.10 Obtaining an Annotation Type From a Specified Object Tag: ANtag2atype

The **ANtag2atype** routine returns theannotation type (prefaced by `AN_`) corresponding to a given object tag (prefaced by `DFTAG_`).

Return values are `AN_FILE_LABEL`, `AN_FILE_DESC`, `AN_DATA_LABEL`, `AN_DATA_DESC`.

TABLE 10W **ANtag2atype Parameter List**

Routine Name (Fortran-77)	Parameter	Data Type		Description
		C	Fortran-77	
ANtag2atype (aftagatype)	ann_type	uint16	integer	Annotation type.