Chapter 11

Single-file Annotations (DFAN API)

11.1 Chapter Overview

The original HDF annotation tools were the single-file tools that constitute the DFAN interface. These tools, which are used to read and write file and data object annotations, are described in this chapter.

Note that there is a multifile annotations interface, called the AN interface, for dealing with annotations. The AN interface supersedes the DFAN interface and is described in Chapter 10, *Annotations (AN API)*.

11.2 The Single-file Annotation Interface

The functions and routines that comprise the single-file annotation interface have names that begin with the string "DFAN" in C; 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 DFAN interface functions.

11.2.1 DFAN Library Routines

These functions are divided into the following categories:

- ***Write routines*** assign a file or object annotation.
- ***Read routines*** retrieve a file or object annotation.
- ***General inquiry routines*** return a list of all labels and reference numbers.
- ***Maintenance routine*** performs cleanup services.

The DFAN interface routines are listed in the following table and are discussed in the subsequent sections of this document.

TABLE 11A **DFAN Library Routines**

Purpose	Functions		Description
	C	FORTRAN-77	
Write	DFANaddfds	daafds	Assigns a file description to a specific file
	DFANaddfid	daafid	Assigns a file label to a specific file
	DFANputdesc	dapdesc	Assigns an object description to a specific data object
	DFANputlabel	daplab	Assigns an object label to a specific data object

Purpose	Functions		Description
	C	FORTRAN-77	
Read	DFANgetdesc	dagdesc	Reads the text of an object description
	DFANgetdesclen	dagdlen	Returns the length of an object description
	DFANgetfds	dagfds	Reads the text of a file description
	DFANgetfdslen	dagfdsl	Returns the length of a file description
	DFANgetfid	dagfid	Reads the text of a file label
	DFANgetfidlen	dagfidl	Returns the length of a file label
	DFANgetlabel	daglab	Reads the text of an object label
	DFANgetlablen	dagllen	Returns the length of an object label
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	Clears the internal tables and structures used by the DFAN interface

11.2.2 Tags in the Annotation Interface

Table 11B lists the annotation tags defined in HDF versions 2.0, 3.0, and 4.0. Newly-defined tag names in each version are bolded. For a more complete list of tags, refer to the *HDF Specification and Developer's Guide*.

TABLE 11B

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

11.3 Programming Model for the DFAN Interface

There are two general programming models for the DFAN interface; the first programming model addresses file annotation while 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.

11.4 Writing Annotations

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

11.4.1 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** are further defined in (See Table 11C on page 354.).

11.4.2 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. The parameter `description` can contain any sequence of ASCII characters and is not limited to a single string (e.g., a carriage return may appear anywhere in the description). The `desc_length` parameter specifies the length of the description.

The parameters of **DFANaddfds** are defined in Table 11C.

TABLE 11C

DFANaddfid and DFANaddfds Parameter List

Routine Name [Return Value] (FORTRAN-77)	Parameter	Parameter Type		Description
		C	FORTRAN-77	
DFANaddfid [intn] **(daafid)**	file_id	int32	integer	File identifier
	label	char *	character*(*)	File label string
DFANaddfds [intn] **(daafds)**	file_id	int32	integer	File identifier
	description	char *	character*(*)	File description string
	desc_length	int32	integer	Length of the description in bytes

EXAMPLE 1.

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
```

11.4.3 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 11D.

11.4.4 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 Table 11D.

TABLE 11D **DFANputlabel and DFANputdesc Parameter List**

Routine Name [Return Value] **(FORTRAN-77)**	**Parameter**	**Parameter Type**		**Description**
		C	**FORTRAN-77**	
DFANputlabel [intn] **(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 [int] **(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
	description	char *	character*(*)	Object description string
	desc_len	int32	integer	Length of the description in bytes

EXAMPLE 2.

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
```

11.5 Reading Annotations

The DFAN interface provides several functions for reading file and data object annotations, which are described below.

11.5.1 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 determine whether 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, *HDF Fundamentals*.

The parameters of **DFANgetfidlen** and **DFANgetfid** are described in Table 11E.

11.5.2 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. As with **DFANgetfidlen**, you must call **DFANgetfds** between calls to **DFANgetfdslen** to get the lengths of successive file descriptions.

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 `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 11E

DFANgetfidlen, DFANgetfid, DFANgetfdslen, and DFANgetfds Parameter List

Routine Name [Return Value] (FORTRAN-77)	Parameter	Parameter Type		Description
		C	FORTRAN-77	
DFANgetfidlen [int32] **(dagfidl)**	file_id	int32	integer	File identifier
	isfirst	intn	integer	Location of the next annotation
DFANgetfid [int32] **(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 [int32] **(dagfdsl)**	file_id	int32	integer	File identifier
	isfirst	intn	integer	Location of the next annotation
DFANgetfds [int32] **(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 3.

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
```

11.5.3 Reading an Object Label: DFANgetlablen and DFANgetlabel

The DFAN programming model for reading a data 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**. **DFANgetlabel** 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. `label_length` can be shorter than the full length of the label in the file, but if so, 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.

11.5.4 Reading an Object Description: DFANgetdesclen and DFANgetdesc

The DFAN programming model for reading a data 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 11F **DFANgetlablen, DFANgetlabel, DFANgetdesc and DFANgetdesclen Parameter List**

Routine Name [Return Value] (FORTRAN-77)	Parameter	Parameter Type		Description
		C	FORTRAN-77	
DFANgetlablen [int32] **(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 [intn] **(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 [int32] **(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 [intn] **(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 4.

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**.

11.6 Maintenance Routines

The DFAN interface provides one function for interface maintenance, **DFANclear**.

11.6.1 Clearing the DFAN Interface Internal Structures and Settings: DFANclear

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

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

DFANclear returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-1`) otherwise. **DFANclear** takes no parameters, as described in the following table.

TABLE 11G **DFANclear Parameter List**

Routine Name [Return Value] **(FORTRAN-77)**	**Parameter**	**Parameter Type**		**Description**
		C	**FORTRAN-77**	
DFANclear [intn] **(daclear)**	None	None	None	None

11.7 Determining Reference Numbers

It is advisable to check the reference number before attempting to assign an object annotation, as the overwriting of reference numbers is not 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.

11.7.1 Determining a Reference Number for the Last Object Accessed: DF*lastref and DF*writeref

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 11H.) 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.

The **DF*lastref** and **DF*writeref** routines are further described in the following table.

TABLE 11H **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
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

11.7.2 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, *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 contents of `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 11I

DFANlablist Parameter List

Routine Name [Return Value] (FORTRAN-77)	Parameter	Parameter Type		Description
		C	FORTRAN-77	
DFANlablist [int] **(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 5.

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
```

11.7.3 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, *HDF Fundamentals*.