* Error Reporting

# Chapter Overview

This chapter describes the main error reporting routines designed for general HDF use and the types of errors handled by the error reporting API and the general structure of the API.

## The HDF Error Reporting API

The HDF error reporting API consists of routines that query error stack information, the names of which are prefaced by “HE”. They are described briefly in Table 13A. Some are primarily for use by HDF developers while others are available to HDF users. In this chapter, three error reporting functions are covered: HEprint, HEvalue and HEstring. Note that only one C error reporting routine has a FORTRAN-77 counterpart: heprnt/heprntf (heprntf is the newer function, supported on all platforms; heprnt is the original function, supported on non-Microsoft Windows platforms).

* Error Reporting Routine List

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Routine Name | | Description |
| C | FORTRAN-77 |
| Error Reporting | HEprint | heprnt heprntf | Prints the errors on the error stack to a specified file. |
| HEstring | hestringf | Returns the error message associated with an error code. |
| HEvalue | None | Returns the nth most recent error reported. |

## Error Reporting in HDF

Most HDF error reporting routines return FAIL (or -1) if the operation is successful and SUCCEED (or 0) otherwise. Each time a FAIL code is returned one or more error codes are pushed onto the error code stack. The following pseudo-code will demonstrate the two methods commonly used to access and print the contents of this stack.

if (<general HDF function() >= FAIL) {

<HDF error reporting API routines>

}

OR

status = <general HDF function( );

if (status == FAIL) {

<HDF error reporting API routines>

}

A list of error codes is included at the end of this chapter.

### Writing Errors to a File: HEprint

HEprint writes the errors on the stack to the specified file. There are four sections of an HEprint error report:

* A description of the error.
* The routine in which the error was detected.
* The source file in which the error was detected.
* The line number in which the error was detected.

The syntax for HEprint is as follows:

C: HEprint(stream, level);

FORTRAN: status = heprnt(level)

The stream parameter is a UNIX file handle indicating the output stream the error information will be written to. The level parameter specifies the amount of error information to report. In FORTRAN-77, heprnt (supported on non-Microsoft Windows platforms) always writes to the standard error stream, or stderr; therefore the only parameter is level. To facilitate Microsoft Windows support, a newer function heprntf (supported on all platforms) requires two parameters, filename to identify the file to which the error information is to be written and level.

Errors are written in sequential order starting from the bottom of the stack. Consequently, specifying a level parameter value of 1 will write the first error that occurred, or the first error pushed onto the stack. Specifying a level parameter of value 0 will write all errors on the stack to the specified file. For example, the following C code will write all errors on the stack to the file named “errors”.

f = fopen("errors", "w");

HEprint(f, 0);

As an example of the output of HEprint, suppose an attempt is made to open an nonexistent file with Hopen. Calling HEprint(stdout, 0) or heprnt(0) will produce the following output:

HDF error: <error opening file>

Detected in Hopen() [hfile.c line 305]

### Returning the Code of the Nth Most Recent Error: HEvalue

HEvalue returns the error code for the nth most recent error and is only available as a C routine. The level parameter specifies the number of errors to regress from the top of the error stack, i.e., HEvalue(1) will return the error code at the top of the stack. Refer to (See Table 13B on page 440) for a complete list of HDF4 error codes.

The syntax for HEvalue is as follows:

C: status = HEvalue(level);

### Returning the Description of an Error Code: HEstring/hestringf

HEstring returns the error description associated with the error code specified by the error\_code parameter as a character string.

The syntax for HEstring is as follows:

C: error\_message = HEstring(error\_code);

FORTRAN: status = hestringf(error\_code, error\_message)

### Clearing the error stack: HEclear

HEclear clears all information on reported errors from the error stack and is only available as a C routine. The syntax for HEclear is as follows:

C: status = HEclear();

Note that every HDF4 API calls HEclear to clear the error stack.

* Writing Errors to a Console Window

The following C code fragment will copy errors from the stack to a console window.

C:

#include "hdf.h"

main( )

{

int32 i, e;

const char \*str;

...

i = 0;

while ((e = HEvalue(i)) != DFE\_NONE) {

str = HEstring(e);

<device-specific code to print the string to a console>

i++

...

}

* HDF Error Codes

|  |  |
| --- | --- |
| Error Code | Code Definition |
| DFE\_NONE | No error. |
| DFE\_FNF | File not found. |
| DFE\_DENIED | Access to file denied. |
| DFE\_ALROPEN | File already open. |
| DFE\_TOOMANY | Too many AID's or files open. |
| DFE\_BADNAME | Bad file name on open. |
| DFE\_BADACC | Bad file access mode. |
| DFE\_BADOPEN | Miscellaneous open error. |
| DFE\_NOTOPEN | File can't be closed because it hasn’t been opened. |
| DFE\_CANTCLOSE | fclose error |
| DFE\_READERROR | Read error. |
| DFE\_WRITEERROR | Write error. |
| DFE\_SEEKERROR | Seek error. |
| DFE\_RDONLY | File is read only. |
| DFE\_BADSEEK | Attempt to seek past end of element. |
| DFE\_PUTELEM | Hputelement error. |
| DFE\_GETELEM | Hgetelement error. |
| DFE\_CANTLINK | Cannot initialize link information. |
| DFE\_CANTSYNC | Cannot synchronize memory with file. |
| DFE\_BADGROUP | Error from DFdiread in opening a group. |
| DFE\_GROUPSETUP | Error from DFdisetup in opening a group. |
| DFE\_PUTGROUP | Error on putting a tag/reference number pair into a group. |
| DFE\_GROUPWRITE | Error when writing group contents. |
| DFE\_DFNULL | Data file reference is a null pointer. |
| DFE\_ILLTYPE | Data file contains an illegal type: internal error. |
| DFE\_BADDDLIST | The DD list is non-existent: internal error. |
| DFE\_NOTDFFILE | The current file is not an HDF file and it is not zero length. |
| DFE\_SEEDTWICE | The DD list already seeded: internal error. |
| DFE\_NOSUCHTAG | No such tag in the file: search failed. |
| DFE\_NOFREEDD | There are no free DDs left: internal error. |
| DFE\_BADTAG | Illegal WILDCARD tag. |
| DFE\_BADREF | Illegal WILDCARD reference number. |
| DFE\_NOMATCH | No DDs (or no more DDs) that match the specified tag/reference number pair. |
| DFE\_NOTINSET | Warning: Set contained unknown tag. Ignored. |
| DFE\_BADOFFSET | Illegal offset specified. |
| DFE\_CORRUPT | File is corrupted. |
| DFE\_NOREF | No more reference numbers are available. |
| DFE\_DUPDD | The new tag/reference number pair has been allocated. |
| DFE\_CANTMOD | Old element doesn’t exist. Cannot modify. |
| DFE\_DIFFFILES | Attempt to merge objects in different files. |
| DFE\_BADAID | An invalid AID was received. |
| DFE\_OPENAID | Active AIDs still exist. |
| DFE\_CANTFLUSH | Cannot flush DD back to file. |
| DFE\_CANTUPDATE | Cannot update the DD block. |
| DFE\_CANTHASH | Cannot add a DD to the hash table. |
| DFE\_CANTDELDD | Cannot delete a DD in the file. |
| DFE\_CANTDELHASH | Cannot delete a DD from the hash table. |
| DFE\_CANTACCESS | Cannot access specified tag/reference number pair. |
| DFE\_CANTENDACCESS | Cannot end access to data element. |
| DFE\_TABLEFULL | Access table is full. |
| DFE\_NOTINTABLE | Cannot find element in table. |
| DFE\_UNSUPPORTED | Feature not currently supported. |
| DFE\_NOSPACE | malloc failed. |
| DFE\_BADCALL | Routine calls were in the wrong order. |
| DFE\_BADPTR | NULL pointer argument was specified. |
| DFE\_BADLEN | Invalid length was specified. |
| DFE\_NOTENOUGH | Not enough space for the data. |
| DFE\_NOVALS | Values were not available. |
| DFE\_ARGS | Invalid arguments passed to the routine. |
| DFE\_INTERNAL | Serious internal error. |
| DFE\_NORESET | Too late to modify this value. |
| DFE\_GENAPP | Generic application level error. |
| DFE\_UNINIT | Interface was not initialized correctly. |
| DFE\_CANTINIT | Cannot initialize the interface the operation requires. |
| DFE\_CANTSHUTDOWN | Cannot shut down the interface the operation requires. |
| DFE\_BADDIM | Negative number of dimensions, or zero dimensions, was specified. |
| DFE\_BADFP | File contained an illegal floating point number. |
| DFE\_BADDATATYPE | Unknown or unavailable data type was specified. |
| DFE\_BADMCTYPE | Unknown or unavailable machine type was specified. |
| DFE\_BADNUMTYPE | Unknown or unavailable number type was specified. |
| DFE\_BADORDER | Unknown or illegal array order was specified. |
| DFE\_RANGE | Improper range for attempted access. |
| DFE\_BADCONV | Invalid data type conversion was specified. |
| DFE\_BADTYPE | Incompatible types were specified. |
| DFE\_BADSCHEME | Unknown compression scheme was specified. |
| DFE\_BADMODEL | Invalid compression model was specified. |
| DFE\_BADCODER | Invalid compression encoder was specified. |
| DFE\_MODEL | Error in the modeling layer of the compression operation. |
| DFE\_CODER | Error in the encoding layer of the compression operation. |
| DFE\_CINIT | Error in encoding initialization. |
| DFE\_CDECODE | Error in decoding compressed data. |
| DFE\_CENCODE | Error in encoding compressed data. |
| DFE\_CTERM | Error in encoding termination. |
| DFE\_CSEEK | Error seeking in an encoded data set. |
| DFE\_MINIT | Error in modeling initialization. |
| DFE\_COMPINFO | Invalid compression header. |
| DFE\_CANTCOMP | Cannot compress an object. |
| DFE\_CANTDECOMP | Cannot decompress an object. |
| DFE\_NOENCODER | Encoder not available. |
| DFE\_NOSZLIB | SZIP library not available. |
| DFE\_COMPVERSION | Version error from zlib  Note: when Z\_VERSION\_ERROR (-6) returned from zlib. |
| DFE\_READCOMP | Error in reading compressed data.  Note: when one of the following error codes returned from zlib:  Z\_ERRNO (-1)  Z\_STREAM\_ERROR (-2)  Z\_DATA\_ERROR (-3)  Z\_MEM\_ERROR (-4)  Z\_BUF\_ERROR (-5) |
| DFE\_NODIM | A dimension record was not associated with the image. |
| DFE\_BADRIG | Error processing a RIG. |
| DFE\_RINOTFOUND | Cannot find raster image. |
| DFE\_BADATTR | Invalid attribute. |
| DFE\_BADTABLE | The nsdg table has incorrect information. |
| DFE\_BADSDG | Error in processing an SDG. |
| DFE\_BADNDG | Error in processing an NDG. |
| DFE\_VGSIZE | Too many elements in the vgroup. |
| DFE\_VTAB | Element not in vtab[]. |
| DFE\_CANTADDELEM | Cannot add the tag/reference number pair to the vgroup. |
| DFE\_BADVGNAME | Cannot set the vgroup name. |
| DFE\_BADVGCLASS | Cannot set the vgroup class. |
| DFE\_BADFIELDS | Invalid fields string passed to vset routine. |
| DFE\_NOVS | Cannot find the vset in the file. |
| DFE\_SYMSIZE | Too many symbols in the users table. |
| DFE\_BADATTACH | Cannot write to a previously attached vdata. |
| DFE\_BADVSNAME | Cannot set the vdata name. |
| DFE\_BADVSCLASS | Cannot set the vdata class. |
| DFE\_VSWRITE | Error writing to the vdata. |
| DFE\_VSREAD | Error reading from the vdata. |
| DFE\_BADVH | Error in the vdata header. |
| DFE\_VSCANTCREATE | Cannot create the vdata. |
| DFE\_VGCANTCREATE | Cannot create the vgroup. |
| DFE\_CANTATTACH | Cannot attach to a vdata or vset. |
| DFE\_CANTDETACH | Cannot detach a vdata or vset with write access. |
| DFE\_BITREAD | A bit read error occurred. |
| DFE\_BITWRITE | A bit write error occurred. |
| DFE\_BITSEEK | A bit seek error occurred. |
| DFE\_TBBTINS | Failed to insert the element into tree. |
| DFE\_BVNEW | Failed to create a bit vector. |
| DFE\_BVSET | Failed when setting a bit in a bit vector. |
| DFE\_BVGET | Failed when getting a bit in a bit vector. |
| DFE\_BVFIND | Failed when finding a bit in a bit vector. |