Chapter 13

Error Reporting

13.1 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.

13.2 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: **HEprint**.

TABLE 13A **Error Reporting Routine List**

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

13.3 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.

13.3.1 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:

1. A description of the error.
2. The routine in which the error was detected.
3. The source file in which the error was detected.
4. 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, **HEprint** always writes to the standard error stream, or `stderr`, therefore the only parameter is `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]
```

13.3.2 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 `error_stack_offset` parameter specifies the number of errors to regress from the top of the error stack.

The syntax for **HEvalue** is as follows:

C: `status = HEvalue(error_stack_offset);`

13.3.3 Returning the Description of an Error Code: HEstring

HEstring returns the error description associated with the error code specified by the `error_code` parameter as a character string. As with **HEvalue**, **HEstring** is only available in the C HDF library.

The syntax for **HEstring** is as follows:

C:
```
error_string = HEstring(error_code);
```

EXAMPLE 1.

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++

...

}
```

TABLE 13B

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.

Error Code	Code Definition
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.

Error Code	Code Definition
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_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.

Error Code	Code Definition
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.