

# HEclear

---

## HEclear

VOID HEclear( )

<b>Purpose</b>	Clears all information on reported errors from the error stack.
<b>Return value</b>	None.
<b>Description</b>	<b>HEpush</b> creates an error stack. <b>HEclear</b> is then used to clear this stack after any errors are processed.

**HEprint/heprnt**

VOID HEprint(FILE \**stream*, int32 *level*)

*stream*           IN:   Stream to print error message to  
*level*            IN:   Level of error stack to print

**Purpose**           Prints information to the error stack.

**Return value**   None.

**Description**    If *level* is 0, all of the errors currently on the error stack are printed. Output from this function is sent to the file pointed to by *stream*.

The following information is printed: the ASCII description of the error, the reporting routine, the reporting routine as source file name, and the line at which the error was reported. If the programmer has supplied extra information by means of **HEreport**, this information is printed as well.

The Fortran-77 routine uses one less parameter than the C routine because it doesn't allow the user to specify the print stream. Instead, it always prints to `stdout`

**FORTTRAN**       subroutine heprnt(*level*)  
  
                  integer *level*

## HEpush

VOID HEpush(int16 *error\_code*, char \**funct\_name*, char \**file\_name*, intn *line*)

<i>error_code</i>	IN:	HDF error code corresponding to the error
<i>funct_name</i>	IN:	Name of function in which the error occurred
<i>file_name</i>	IN:	Name of file in which the error occurred
<i>line</i>	IN:	Line number in the file that error occurred

**Purpose** Pushes a new error onto the error stack.

**Return value** None.

**Description** **HEpush** pushes the file name, function name, line number, and generic description of the error onto the error stack. **HEreport** can then be used to give a more case-specific description of the error.

If the stack is full, the error is ignored. **HEpush** assumes that the character strings *funct\_name* and *file\_name* are in semi-permanent storage, so only pointers to the strings are saved.

**HEreport**

VOID HEreport(char \**format*, ... )

*format*            IN:    Output string specification

**Purpose**            Adds a text string to the description of the most-recently-reported error (only one text string per error).

**Return value**      None

**Description**       **HEpush** places on the error stack the file name, function name, line number, and a generic description of the error type. **HEreport** can then be used to give a more case-specific description of the error. Only one additional annotation can be attached to each error report.

The format argument must conform to the string specification requirements of `printf`

**Example**

```
char *FUNC = "Hclose";

if (file_rec->attach > 0) {
    file_rec->refcount++;
    ERROR(DFE_OPENAID);
    HEreport("There are still %d active aids attached",
            file_rec->attach);
    return FAIL;
}
```

# HEstring

---

## HEstring

char \*HEstring(int16 *error\_code*)

*error\_code*      IN:    HDF error code

**Purpose**               Returns the error message associated with specified error code.

**Return value**       Returns a pointer to a string associated with the error code if successful.

**Description**       Returns a text description of the given error code. These strings are statically declared and should not be deallocated from memory (using the `free` routine) by the user. If a defined text description cannot be found a generic default message is returned.

**HEvalue**

int16 HEvalue(int32 *level*)

*level*                    IN:    Level of the error stack to be returned

**Purpose**                Returns an error from the specified level of the error stack.

**Return value**        The error code if successful for DFE\_NONE otherwise.