

**HXcreate**

```
int32 HXcreate(int32 file_id, uint16 tag, uint16 ref, char *extern_name, int32 offset, int32 length)
```

<i>file_id</i>	IN: File identifier returned by <b>Hopen</b>
<i>tag</i>	IN: Tag of the external element
<i>ref</i>	IN: Reference number of the external element
<i>extern_name</i>	IN: Name of the external file
<i>offset</i>	IN: Offset, in bytes, of the first byte of the external element
<i>length</i>	IN: Length of the external element

<b>Purpose</b>	Creates an external element.
<b>Return value</b>	Returns the access identifier for the external element if successful and <code>FAIL</code> (or <code>-1</code> ) otherwise.
<b>Description</b>	An external element is a special data element stored in a file other than the one that contains its data descriptor. <b>HXcreate</b> can create new external elements, move existing data elements from and HDF file to an external file, or incorporate data from an external file into an HDF file.

**HXcreate** creates a new external element if given an unused tag/reference number pair and the file name of a nonexistent file. Specify *offset* as the number of bytes between the beginning of the external file and the beginning of the external element. After creating the new external element, use **Hwrite** to write data to it.

**HXcreate** moves a data element from an HDF file to an external file if given a preexisting tag/reference number pair. During this operation, the original data element is deleted from the HDF file and copied to the external file. Any data located at the specified offset in the external file is overwritten. Specify *offset* as if creating a new external element. The *length* argument is ignored if the object already exists in the file - it's set to the actual length of the file.

**HXcreate** incorporates data from an external file into an HDF file if given and new tag/reference number pair and the file name of an existing external file. This operation does not modify the external file, it simply creates a data descriptor in the specified HDF file that points to the external data. The *offset* and *length* arguments should reflect the location and length of the data in the external file.

Moving existing data elements to external files will not automatically close open access identifiers. Open access identifiers prevent a file from closing, therefore it is important to use **Hendaccess** to remove any open identifiers before attempting to close a file.

Distributing a single data element over several external files is not supported.

**HXsetcreatedir/hxiscdir**

intn HXsetcreatedir(const char \**dir*)

*dir*                      IN:    Target directory of the external file to be written

**Purpose**                      Initializes the directory environment variable, identifying the location of the external file to be written.

**Return value**               Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

**Description**               The contents of *dir* is copied into the private memory of the HDF library. If *dir* is NULL, the directory variable is unset. If **HXsetcreatedir** encounters an error condition, the directory variable is not changed. When a new external element is created (via the **HXcreate** or **SDsetexternal** routines), the HDF library accesses the external file just like the **open** call by default. Refer to the Reference Manula page on **HXcreate** for a description of when a new or an old file should be opened.

Users may override the default action by calling **HXsetcreatedir** or by defining the environment variable \$HDFEXTCREATEDIR. The HDF library will access the external file in the directory according to the environment variable setting. The precedence is **HXsetcreatedir**, then \$HDXEXTDIR, in the manner of **open**.

Note that the above override does not apply to absolute pathnames - i.e., filenames starting with a forward slash. HDF will access the absolute pathname without change. Also note that **HXsetcreatedir** and \$HDFEXTCREATEDIR are not symmetrical to **HXsetdir** and \$HDFEXTDIR. The former pair permits only single directory values and is used to compose the filename for access. The later pair permits multiple directory values which are used for searching an existing file.

**FORTTRAN**                      integer function hxiscdir(*dir*)  
  
integer *dir*

# HXsetdir/hxisdir

---

## HXsetdir/hxisdir

intn HXsetdir(const char \**dir*)

*dir*                      IN:    Target directory of the external file to be located

**Purpose**                      Initializes the directory environment variable, identifying the location of the external file to be located.

**Return value**               Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

**Description**               **HXsetdir** sets the directory variable for locating an external file according to *dir* which may contain multiple directories separated by vertical bars (e.g., "dir1|dir2"). The content of *dir* is copied into the private memory of the HDF library. If *dir* is NULL, the directory variable is unset.

If **HXsetdir** encounters any error, the directory variable is not changed. By default, the HDF library locates the external file just like the **open** call. It also searches for the external file in the directories specified by the user environment variable \$HDFEXTDIR, if defined, and the directory variable set by **HXsetdir**. The searching precedence is directory variable, if set, then \$HDXEXTDIR, then in the manner of **open**.

The searching differs if the external filename is an absolute pathname - i.e., starting with a forward slash. HDF will try **open** first. If **open** fails and if \$HDFEXTDIR is defined or the directory variable is set via **HXsetdir**, HDF will remove all directory components of the absolute pathname (e.g., "/usr/groupA/projectB/Data001" becomes "Data001") and search for that filename with the strategy described in the previous paragraph.

FORTRAN                      integer function hxisdir(*dir*)  
  
integer *dir*