

HLconvert

intn HLconvert(int32 *h_id*, int32 *block_length*, int32 *n_blocks*)

<i>h_id</i>	IN: Standard access element identifier
<i>block_length</i>	IN: Standard length of a linked block
<i>n_blocks</i>	IN: Number of linked list objects in each block header

Purpose Converts a standard access element into a linked block special data element that can be easily appended.

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

Description If the standard-access element already exists, it is promoted to being a linked-block element, otherwise a new element is created.

All of the blocks of the linked list are the same size (specified by *block_length*) except for the first one which retains its size at the time **HLcreate** was called.

This routine is similar to **HLcreate** but is used to convert an existing standard access element into a linked-block element "in-place". This is done for convenience and ease-of-use in many HDF library routines, but it is allowable for user-level code to do this also.

Appropriate values for *n_blocks* and *block_length* are very data and application dependent.

HLcreate

```
int32 HLcreate(int32 file_id, uint16 tag, uint16 ref, int32 block_len, int32 n_blocks)
```

<i>file_id</i>	IN: File identifier returned by Hopen
<i>tag</i>	IN: Tag of the new data descriptor
<i>ref</i>	IN: Reference number of the new data descriptor
<i>block_len</i>	IN: Length of blocks to be created
<i>n_blocks</i>	IN: Number of blocks to be created per linked-block record

Purpose Creates a linked-block special data element.

Return value Returns an access identifier if successful and `FAIL` (or `-1`) otherwise.

Description Appending data to existing data objects is a problem in early versions of the HDF library because objects are required to be stored contiguously. When appending, the HDF library deleted the existing element and moved it to the end of the file. HDF version 3.2 added the concept of linked block elements which allow unlimited appending without deleting or copying existing data.

HLcreate creates a new linked-block element if given an unused tag and reference number. If an existing tag and reference number are used, **HLcreate** promotes the existing data element to a linked block element. To create a linked data element, specify the number of data blocks for the element in *n_blocks* and their length in *block_len*. All data blocks must be the same length unless the element was promoted from a non-linked-block data element. In this case, all but the original data block must conform to the length specification.

HLcreate creates linked-block elements by inserting a linked-block structure between a data descriptor and its associated data element. The linked-block structure is used to keep track of the different blocks of data appended to the same data object. All data blocks associated with a new linked-block element are stored at the same location. If a linked-block element is promoted from a standard data element, the block containing the original data remains in its original position with the remaining data blocks stored together at a different file location. This non-contiguous configuration is known to cause problems with **Hoffset** and **Hlength** which assume that data in a data element is stored contiguously.

Promoting existing data elements to linked-block elements will not automatically close opened 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.