H5DO_WRITE_CHUNK

Expand all Collapse all

- Jump to ...
- Summary
- Description
- Example
- Switch language ...
- C
- C++
- FORTRAN
- JAVA

H5DO_WRITE_CHUNK

Writes a raw data chunk from a buffer directly to a dataset in a file (DEPRECATED)

> *This function was deprecated in favor of the function H5D_WRITE_CHUNK as of HDF5-1.10.3. The functionality of H5DO_WRITE_CHUNK was moved to H5D_WRITE_CHUNK. For compatibility, this API call has been left as a stub which simply calls H5D_WRITE_CHUNK. New code should use H5D_WRITE_CHUNK.*

Procedure:

H5DO_WRITE_CHUNK (dset_id, dxpl_id, filter_mask, offset, data_size, buf)

Signature:

```
herr_t H5DOwrite_chunk( hid_t dset_id, hid_t dxpl_id, uint32_t filter_mask, hsize_t *offset, size_t
data_size, const void *buf )
```

Parameters:

hid_t `dset_id`	IN: Identifier for the dataset to write to
hid_t `dxpl_id`	IN: Transfer property list identifier for this I/O operation
uint32_t `filter_mask`	IN: Mask for identifying the filters in use
hsize_t `*offset`	IN: Logical position of the chunk's first element in the dataspace
size_t `data_size`	IN: Size of the actual data to be written in bytes

const void *buf	IN: Buffer containing data to be written to the chunk

Description:

H5DO_WRITE_CHUNK writes a raw data chunk as specified by its logical offset in a chunked dataset dset_id from the application memory buffer buf to the dataset in the file. Typically, the data in buf is preprocessed in memory by a custom transformation, such as compression. The chunk will bypass the library's internal data transfer pipeline, including filters, and will be written directly to the file.

dxpl_id is a data transfer property list identifier.

filter_mask is a mask providing a record of which filters are used with the chunk. The default value of the mask is zero (0), indicating that all enabled filters are applied. A filter is skipped if the bit corresponding to the filter's position in the pipeline (0 position < 32) is turned on. This mask is saved with the chunk in the file.

offset is an array specifying the logical position of the first element of the chunk in the dataset's dataspace. The length of the offset array must equal the number of dimensions, or rank, of the dataspace. The values in offset must not exceed the dimension limits and must specify a point that falls on a dataset chunk boundary.

data_size is the size in bytes of the chunk, representing the number of bytes to be read from the buffer buf. If the data chunk has been precompressed, data_size should be the size of the compressed data.

buf is the memory buffer containing data to be written to the chunk in the file.

> Exercise caution when using H5DO_READ_CHUNK and H5DO_WRITE_CHUNK, as they read and write data chunks directly in a file. H5DO_WRITE_CHUNK bypasses hyperslab selection, the conversion of data from one datatype to another, and the filter pipeline to write the chunk. Developers should have experience with these processes before using this function. Please see Using the Direct Chunk Write Function for more information.
>
> Also note that H5DO_READ_CHUNK and H5DO_WRITE_CHUNK are not supported under parallel and do not support variable length types.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

Example:

The following code illustrates the use of H5DOwrite_chunk to write an entire dataset, chunk by chunk:

```
#include <zlib.h>
#include <math.h>
#define DEFLATE_SIZE_ADJUST(s) (ceil(((double)(s))*1.001)+12)
                :
                :
size_t       buf_size = CHUNK_NX*CHUNK_NY*sizeof(int);
const Bytef *z_src = (const Bytef*)(direct_buf);
Bytef       *z_dst;             /* Destination buffer            */
uLongf       z_dst_nbytes = (uLongf)DEFLATE_SIZE_ADJUST(buf_size);
uLong        z_src_nbytes = (uLong)buf_size;
int          aggression = 9;    /* Compression aggression setting */
uint32_t     filter_mask = 0;
size_t       buf_size = CHUNK_NX*CHUNK_NY*sizeof(int);

/* Create the data space */
if((dataspace = H5Screate_simple(RANK, dims, maxdims)) < 0)
    goto error;

/* Create a new file */
if((file = H5Fcreate(FILE_NAME5, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT)) < 0)
    goto error;

/* Modify dataset creation properties, i.e. enable chunking and compression */
if((cparms = H5Pcreate(H5P_DATASET_CREATE)) < 0)
```

```
    goto error;

if((status = H5Pset_chunk( cparms, RANK, chunk_dims)) < 0)
    goto error;

if((status = H5Pset_deflate( cparms, aggression)) < 0)
    goto error;

/* Create a new dataset within the file using cparms creation properties */
if((dset_id = H5Dcreate2(file, DATASETNAME, H5T_NATIVE_INT, dataspace, H5P_DEFAULT,
                         cparms, H5P_DEFAULT)) < 0)
    goto error;

/* Initialize data for one chunk */
for(i = n = 0; i < CHUNK_NX; i++)
    for(j = 0; j < CHUNK_NY; j++)
        direct_buf[i][j] = n++;

/* Allocate output (compressed) buffer */
outbuf = malloc(z_dst_nbytes);
z_dst = (Bytef *)outbuf;

/* Perform compression from the source to the destination buffer */
ret = compress2(z_dst, &z_dst_nbytes, z_src, z_src_nbytes, aggression);

/* Check for various zlib errors */
if(Z_BUF_ERROR == ret) {
   fprintf(stderr, "overflow");
   goto error;
} else if(Z_MEM_ERROR == ret) {
   fprintf(stderr, "deflate memory error");
   goto error;
} else if(Z_OK != ret) {
   fprintf(stderr, "other deflate error");
   goto error;
}

/* Write the compressed chunk data repeatedly to cover all the
 * chunks in the dataset, using the direct write function.     */
for(i=0; i<NX/CHUNK_NX; i++) {
    for(j=0; j<NY/CHUNK_NY; j++) {
        status = H5DOwrite_chunk(dset_id, H5P_DEFAULT, filter_mask,
                                 offset, z_dst_nbytes, outbuf);
        offset[1] += CHUNK_NY;
    }
    offset[0] += CHUNK_NX;
    offset[1] = 0;
}

/* Overwrite the first chunk with uncompressed data.  Set the filter mask to
 * indicate the compression filter is skipped */
filter_mask = 0x00000001;
offset[0] = offset[1] = 0;
if(H5DOwrite_chunk(dset_id, H5P_DEFAULT, filter_mask, offset, buf_size,
             direct_buf) < 0)
   goto error;

/* Read the entire dataset back for data verification converting ints to longs */
if(H5Dread(dataset, H5T_NATIVE_LONG, H5S_ALL, H5S_ALL, H5P_DEFAULT,
             outbuf_long) < 0)
   goto error;

/* Data verification here */
```

```
                    :
                    :
```

History:

Release	Change
1.10.3	Function deprecated in favor of `H5Dwrite_chunk`.
1.8.11	C function introduced in this release.

--- Last Modified: June 03, 2019 | 03:56 PM