H5D_SCATTER

Expand all Collapse all

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

Summary
Description
Example
JAVA
FORTRAN
C++
C

H5D_SCATTER

Scatters data into a selection within a memory buffer

Procedure:

H5D_SCATTER(op, op_data,type_id, dst_space_id,dst_buf)

Signature:

```
herr_t H5Dscatter( H5D_scatter_func_t op, void * op_data, hid_t type_id, hid_t dst_space_id, void *dst_buf
)
```

Parameters:

H5D_scatter_func_t `op`	IN: Callback function which provides data to be scattered
void *`op_data`	IN: User-defined pointer to data required by `op`
hid_t `type_id`	IN: Identifier for the datatype describing the data in both the source and definition buffers This is only used to calculate the element size.
hid_t `dst_space_id`	IN: Identifier for the dataspace describing both the dimensions of the destination buffer and the selection within the destination buffer that data will be scattered to

void *dst_buf	OUT: Destination buffer which the data will be scattered to

Description:

H5D_SCATTER retrieves data from the supplied callback op and scatters it to the supplied buffer dst_buf in a manner similar to data being written to a dataset.

dst_space_id is a dataspace which defines the extent of dst_buf and the selection within it to scatter the data to.

type_id is the datatype of the data to be scattered in both the source and destination buffers.

dst_buf must be at least as large as the number of elements in the extent of dst_space_id times the size in bytes of type_id.

To retrieve the data to be scattered, H5D_SCATTER repeatedly calls op, which should return a valid source buffer, until enough data to fill the selection has been retrieved. The prototype of the callback function op is as follows (as defined in the source code file H5Dpublic.h):

herr_t (*H5D_scatter_func_t)(*const void* ** src_buf/*out*/, *size_t* *src_buf_bytes_used/*out*/, *void* *op_data)

The parameters of this callback function have the following values or meanings:

src_buf	Pointer to the buffer holding the next set of elements to scatter On entry, the value of *src_buf is undefined. The callback function should set *src_buf to point to the next set of elements.
src_buf_bytes_used	Pointer to the number of valid bytes in src_buf On entry, the value of *src_buf_bytes_used is undefined. The callback function should set *src_buf_bytes_used to the number of valid bytes in src_buf. This number must be a multiple of the datatype size.
op_data	User-defined pointer to data required by the callback function A pass-through of the op_data pointer provided with theH5D_SCATTER function call.

The callback function should always return at least one element in src_buf, and must not return more elements than are remaining to be scattered. This function will be repeatedly called until all elements to be scattered have been returned. The callback function should return zero (0) to indicate success, and a negative value to indicate failure.

> **Note for C++ Developers Using C Functions:**
>
> If a C routine that takes a function pointer as an argument is called from within C++ code, the C routine should be returned from normally.
>
> Examples of this kind of routine include callbacks such as H5P_SET_ELINK_CB and H5P_SET_TYPE_CONV_CB and functions such as H5T_CONVERT and H5E_WALK2.
>
> Exiting the routine in its normal fashion allows the HDF5 C library to clean up its work properly. In other words, if the C++ application jumps out of the routine back to the C++ “catch” statement, the HDF5 C library is not given the opportunity to close any temporary data structures that were set up when the routine was called. The C++ application should save some state as the routine is started so that any problem that occurs might be diagnosed.

Returns:

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

Example:

Coming Soon!

History:

Release	Change
1.8.11	C function introduced.

--- Last Modified: December 18, 2018 | 01:35 PM