H5I_REGISTER_TYPE

H5I_REGISTER_TYPE

Creates and returns a new ID type

Procedure:

H5I_REGISTER_TYPE(hash_size, reserved, free_func)

Signature:

```
H5I_type_t H5Iregister_type( size_t hash_size, unsigned reserved, H5I_free_t free_func )
```

Parameters:

size_t `hash_size`	IN: Size of the hash table (in entries) used to store IDs for the new type
unsigned `reserved`	IN: Number of reserved IDs for the new type
H5I_free_t `free_func`	IN: Function used to deallocate space for a single ID

Description:

H5I_REGISTER_TYPE allocates space for a new ID type and returns an identifier for it.

The `hash_size` parameter indicates the minimum size of the hash table used to store IDs in the new type.

The `reserved` parameter indicates the number of IDs in this new type to be reserved. Reserved IDs are valid IDs which are not associated with any storage within the library.

The `free_func` parameter is a function pointer to a function which returns an *herr_t* and accepts a *void **. The purpose of this function is to deallocate memory for a single ID. It will be called by H5I_CLEAR_TYPE and H5I_DESTROY_TYPE on each ID. This function is NOT called by H5I_REMOVE_VERIFY. The *void ** will be the same pointer which was passed in to the H5I_REGISTER function. The `free_func` function should return 0 on success and -1 on failure.

Programming 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 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 the type identifier on success, negative on failure.

Example:

Coming Soon!

--- Last Modified: April 25, 2019 | 12:31 PM