H5I_SEARCH

Expand all Collapse all

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

Summary
Description
Example
JAVA
FORTRAN
C++
C

H5I_SEARCH

Finds the memory referred to by an ID within the given ID type such that some criterion is satisfied

Procedure:

H5I_SEARCH(type, func, key)

Signature:

```
void *H5Isearch( H5I_type_t type, H5I_search_func_t func, void *key )
```

Parameters:

H5I_type_t `type`	IN: The identifier of the type to be searched
H5I_search_func_t `func`	IN: The function defining the search criteria
void *`key`	IN: A key for the search function

Description:

H5I_SEARCH searches through a given ID type to find an object that satisfies the criteria defined by `func`. If such an object is found, the pointer to the memory containing this object is returned. Otherwise, `NULL` is returned. To do this, `func` is called on every member of `type`. The first member to satisfy `func` is returned.

The `type` parameter is the identifier for the ID type which is to be searched. This identifier must have been created by a call to H5I_REGISTER_TYPE.

The parameter `func` is a function pointer to a function which takes three parameters. The first parameter is a *void* * and will be a pointer to the object to be tested. This is the same object that was placed in storage using H5I_REGISTER. The second parameter is a *hid_t* and is the ID of the object to be tested. The last parameter is a *void* *. This is the `key` parameter and can be used however the user finds helpful, or it can be ignored if it is not needed. `func` returns 0 if the object it is testing does not pass its criteria. A non-zero value should be returned if the object does pass its criteria. `H5I_search__func_t` is defined in H5Ipublic.h and is shown below.

```
typedef int (*H5I_search_func_t)(void *obj, hid_t id, void *key);
```

The `key` parameter will be passed to the search function as a parameter. It can be used to further define the search at run-time.

> **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 a pointer to the object which satisfies the search function on success, `NULL` on failure.

Example:

Coming Soon!

--- Last Modified: May 24, 2018 | 04:12 PM