H5L_ITERATE_BY_NAME1

Expand all Collapse all

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

H5L_ITERATE_BY_NAME1

Iterates through links in a group by its name

> *As of HDF5-1.12 this function has been deprecated in favor of the function H5L_ITERATE_BY_NAME2 or the macro H5L_ITERATE_BY_NAME.*

Procedure:

H5L_ITERATE_BY_NAME1 (loc_id, group_name, idx_type, order, idx_p, op, op_data, lapl_id)

Signature:

```
herr_t H5Literate_by_name1 ( hid_t loc_id, const char *group_name, H5_index_t idx_type, H5_iter_order_t
order,
                  hsize_t *idx_p, H5L_iterate1_t op, void *op_data, hid_t lapl_id )
```

Fortran2003:

```
SUBROUTINE h5literate_by_name_f(loc_id, group_name, index_type, &
           order, idx, op, op_data, return_value, hdferr, lapl_id)
    INTEGER(HID_T), INTENT(IN) :: loc_id
    CHARACTER(LEN=*) :: group_name
    INTEGER, INTENT(IN) :: index_type
    INTEGER, INTENT(IN) :: order
    INTEGER(HSIZE_T), INTENT(INOUT) :: idx

    TYPE(C_FUNPTR):: op
    TYPE(C_PTR)   :: op_data

    INTEGER, INTENT(OUT) :: return_value

    INTEGER, INTENT(OUT) :: hdferr
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: lapl_id
```

Programming Note for Fortran Developers:

The integer type of the callback function must match the C integer type. Therefore, for portability, all Fortran callback functions used by h5literate_by_name_f should be declared as INTEGER(KIND=C_INT).

Parameters:

hid_t `loc_id`	IN: Location identifier of subject group; may be a file, group, dataset, named datatype or attribute identifier
const char *`group_name`	IN: Name of subject group
H5_index_t `idx_type`	IN: Type of index which determines the order
H5_iter_order_t `order`	IN: Order within index
hsize_t *`idx_p`	IN: Iteration position at which to start OUT: Position at which an interrupted iteration may be restarted
H5L_iterate1_t `op`	IN: Callback function passing data regarding the link to the calling application
void *`op_data`	IN: User-defined pointer to data required by the application for its processing of the link
hid_t `lapl_id`	IN: Link access property list

Description:

H5L_ITERATE_BY_NAME1 iterates through the links in a group, specified by `loc_id` and `group_name`, in the order of the specified index, `idx_type`, using a user-defined callback routine `op`. H5L_ITERATE_BY_NAME1 does not recursively follow links into subgroups of the specified group.

`idx_type` specifies the index to be used. If the links have not been indexed by the index type, they will first be sorted by that index then the iteration will begin; if the links have been so indexed, the sorting step will be unnecessary, so the iteration may begin more quickly. Valid values include the following:

`H5_INDEX_NAME`	Alpha-numeric index on name
`H5_INDEX_CRT_ORDER`	Index on creation order

`order` specifies the order in which objects are to be inspected along the index specified in `idx_type`. Valid values include the following:

`H5_ITER_INC`	Increasing order
`H5_ITER_DEC`	Decreasing order

H5_ITER_NATIVE	Fastest available order

idx allows an interrupted iteration to be resumed; it is passed in by the application with a starting point and returned by the library with the point at which the iteration stopped.

H5L_ITERATE_BY_NAME1 is not recursive. In particular, if a member of group_name is found to be a group, call it subgroup_a, H5L_ITERATE_BY_NAME1 does not examine the members of subgroup_a. When recursive iteration is required, the application must handle the recursion, explicitly calling H5L_ITERATE_BY_NAME1 on discovered subgroups.

H5L_ITERATE_BY_NAME1 assumes that the membership of the group being iterated over remains unchanged through the iteration; if any of the links in the group change during the iteration, the function's behavior is undefined. Note, however, that objects pointed to by the links can be modified.

H5L_ITERATE_BY_NAME1 is the same as H5G_ITERATE, except that H5G_ITERATE always proceeds in alphanumeric order.

> **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:

On success, returns the return value of the first operator that returns a positive value, or zero if all members were processed with no operator returning non-zero.

On failure, returns a negative value if something goes wrong within the library, or the first negative value returned by an operator.

Example:

History:

Release	Change
1.12.0	Function H5L_ITERATE_BY_NAME was renamed to H5L_ITERATE_BY_NAME1 and deprecated.
1.8.8	Fortran subroutine added.
1.8.0	C function introduced.

--- Last Modified: February 17, 2020 | 08:34 AM