RFC: External Link Traversal Callback

Neil Fortner
Quincey Koziol

This RFC proposes new public routines to allow greater flexibility when traversing external links. These new routines allow the user to define a callback function that can modify the file access property list and access flags used to open the target files of external links. The RFC also proposes new public routines to set the file access flags to use when traversing an external link without using a callback function.

1 Introduction

H5Pset_elink_fapl specifies a file access property list (FAPL) to control application behavior when traversing an external link. While this capability is sufficient in many circumstances, advanced users may need more control over the file access flag than H5Pset_elink_fapl provides. For example, users may also need to specify a file access flag (read only, etc.) to use when opening a file through an external link. This RFC proposes four new public routines, two of which allow users to specify a callback function to call immediately before an external link is traversed. This callback function can modify the FAPL and file access flag used to open target files. The other two public routines allow the user to specify the file access flag to use when opening a target file without using a callback function.

2 Motivation

External links are a powerful tool in HDF5. They provide a great deal of flexibility by allowing path names from the main parent file to refer to objects in different files. The file and/or object referred to by an external link can be quickly changed by simply changing the value of the external link, allowing the same operations to be seamlessly performed on many different datasets in one or more files. The process performing these operations need not have any knowledge of the file the target object resides in.

If any of the potential target files for a single external link need to be opened with a different file driver than others, then the process traversing the external link must currently traverse the path step by step, at each step checking for an external link and setting the FAPL based on the value. If the path might contain more than one external link, then it becomes even more cumbersome.

The user may also wish to change the method of access depending on the parent file or group, or some combination of parent and child names. For example, the user could create a master “read” file, and a master “write” file, and force files opened through the “read” file to be opened in read only mode.

3 Approach

To solve these problems, we are proposing the addition of a callback function that, if defined by the user, will be called immediately before an external link is traversed. This callback function will be passed the parent file name, the group the external link is in, the target file name, the target path name, the file access flag, and the FAPL. The callback function can then adjust both the FAPL and the file access flag, which will then be used to open the target file. These settings will take precedence over the static settings (if any) on the LAPL used to open the external link. The initial values of the FAPL and access flag given to the callback will be those that would otherwise be used to open the target file.

The callback function will be enabled by the new function H5Pset_elink_cb, which takes a function pointer and adds it to a link access property list (LAPL). The value of that function pointer can be retrieved with the new function H5Pget_elink_cb.

We are also proposing new functions to set the file access flag in a LAPL used when traversing an external link. The functions, H5Pset_elink_acc_flags and H5Pget_elink_acc_flags, will provide users who do not wish to use the callback function a simpler way to change the access flag for external link traversal. These functions will also maintain symmetry with H5Pset_elink_fapl and H5P_get_elink_fapl.

Figure 1 on the right illustrates the procedure that will be used by the library to determine the file access flag and FAPL to use when opening the external link target file.

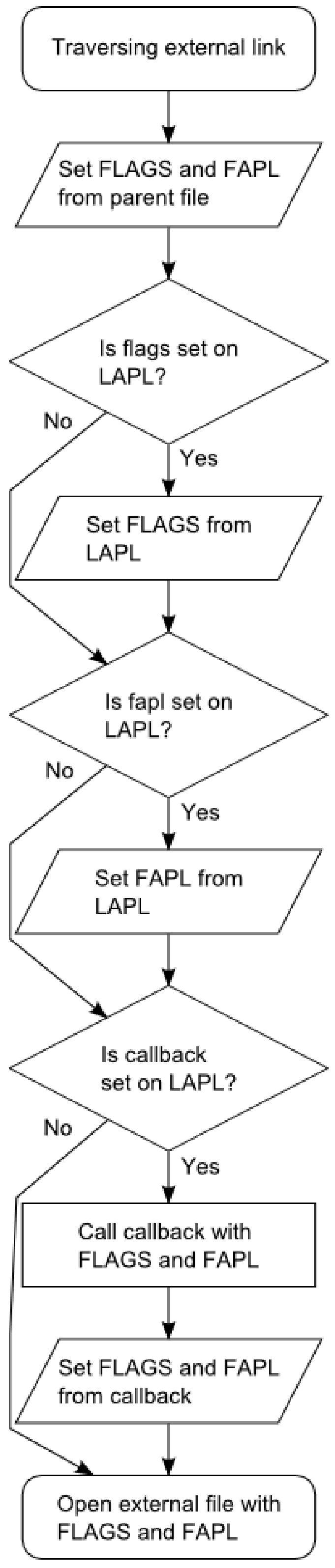

Figure 1: Flowchart for opening an external link.

4 Sample Reference Manual Entries

4.1 H5Pset_elink_cb

Name: H5Pset_elink_cb
Signature:

herr_t `H5Pset_elink_cb`(*hid_t* `lapl_id`, *H5L_elink_traverse_t* `func`, *void* *`op_data`)

Purpose:

Sets the external link traversal callback function.

Description:

`H5Pset_elink_cb` sets the user-defined external link traversal callback function in the link access property list *lapl_id*.

The parameter *op_data* is a pointer to the user-defined input data for the callback function and will be passed through to the callback function.

The callback function *func* may adjust the file access property list and file access flag to use when opening a file through an external link. The callback will be made immediately before opening the target file. The function prototype is as follows:

```
typedef herr_t (*H5L_elink_traverse_t)(const char *parent_file_name,
    const char *parent_group_name, const char *chld_file_name,
    const char *child_object_name, unsigned *acc_flags, hid_t fapl_id,
    void *op_data )
```

parent_file_name and *parent_group_name* specify the parent file and group containing the external link.

child_file_name and *child_object_name* specify the target file and object name.

acc_flags controls the file access flag used to open the target file. This should be set to either `H5F_ACC_RDWR` or `H5F_ACC_RDONLY`. See `H5Fopen` for more information. The initial value of this field will be the flag that would otherwise be used to open the target file as inherited from the parent file, or overridden with `H5Pset_elink_acc_flags`. After making the callback, the flag returned in this parameter will always be used to open the target file.

fapl_id is the identifier of the file access property list used to open the target file. This will initially be a copy of the property list that would otherwise be used to open the target file, as inherited from the parent file, or overridden with `H5Pset_elink_fapl`. After making the callback, this property list, including any changes made by the callbackfunction, will always be used to open the target file.

op_data is a pointer to user-defined input data. This is a pass-through of the value that was passed to `H5Pset_elink_cb`.

The callback function should return 0 if there are no issues and a negative value in case of an error. If the callback function returns a negative value, the external link will not be traversed and an error will be returned.

Parameters:

hid_t `lapl_id`	IN: Link access property list identifier.
H5L_elnk_traverse_t `func`	IN: User-defined external link traversal callback function.
void *`op_data`	IN: User-defined input data for the callback function.

Returns:

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

4.2 H5Pget_elink_cb

Name: H5Pget_elink_cb

Signature:

herr_t `H5Pget_elink_cb`(*hid_t* `lapl_id`, *H5L_elink_traverse_t* *`func`, *void* **`op_data`)

Purpose:

Gets the external link traversal callback function.

Description:

`H5Pget_elink_cb` gets the user-defined external link traversal callback function defined in the link access property list *lapl_id*.

The parameter *op_data* is a pointer to the user-defined input data for the callback function and will be passed through to the callback function.

The callback function *func* may adjust the file access property list and file access flags to use when opening a file through an external link.

See `H5Pset_elink_cb` for more details.

Parameters:

hid_t `lapl_id`	IN: Link access property list identifier.
H5L_elink_traverse_t *`func`	OUT: User-defined external link traversal callback function.
void **`op_data`	OUT: User-defined input data for the callback function.

Returns:

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

4.3 H5Pset_elink_acc_flags

Name: H5Pset_elink_acc_flags

Signature:

herr_t `H5Pset_elink_acc_flags`(*hid_t* `lapl_id`, *unsigned* `flags`)

Purpose:

Sets the external link traversal file access flag.

Description:

`H5Pset_elink_acc_flags` sets the file access flag used to open an external link target file in the link access property list *lapl_id*.

The parameter *flags* contains the access flag for external link traversal. This should be set to either `H5F_ACC_RDWR` or `H5F_ACC_RDONLY`. To remove the setting from *lapl_id*, this should be set to `H5F_ACC_DEFAULT`, which will cause the file access flag setting to be taken from the parent file. See `H5Fopen` for more information.

Parameters:

hid_t `lapl_id`	IN: Identifier of the link access property list.
unsigned `flags`	IN: File access flag for link traversal.

Returns:

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

4.4 H5Pget_elink_acc_flags

Name: H5Pget_elink_acc_flags
Signature:
herr_t `H5Pget_elink_acc_flags`(*hid_t* `lapl_id`, *unsigned* *`flag`)
Purpose:
Gets the external link traversal file access flag.
Description:
`H5Pget_elink_acc_flags` gets the file access flag used to open an external link target file in the link access property list *lapl_id*.

The access flag for external link traversal will be returned in *flags*. If no access flag is set in *lapl_id*, the value of *flags* will be `H5F_ACC_DEFAULT`. See `H5Pset_elink_acc_flags` and `H5Fopen` for more information.

Parameters:

hid_t `lapl_id`	IN: Link access property list identifier.
unsigned *`flags`	OUT: File access flag for link traversal.

Returns:

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

5 Example

This example illustrates the use of a simple callback function for external link traversal. The callback matches the target file name against a set of known file names, and sets the file access properties and file access flags accordingly. The main function also sets the LAPL to default to read only access if it is not set by the callback function.

```
#define FILE0 …

#define FILE1 …

#define FILE2 …

#define OBJ_NAME …
```

```
herr_t elink_callback(const char *parent_file, const char *parent_group, const char
*child_file, const char *child_obj, unsigned *flags, hid_t fapl, void *op_data)
{
      /* Check whether the target file is FILE1 */
      if (!strcmp(FILE1, child_file)) {

            /* Use default access flags */

            /* Use the core file driver, with backing store enabled */
            H5Pset_fapl_core(fapl, 1024*1024, 1);

      /* Check whether the target file is FILE2 */
      } else if (!strcmp(FILE2, child_file)) {

            /* Use read-write acces */
            *flags = H5F_ACC_RDWR;

            /* Force 8-byte alignment in FILE2 */
            H5Pset_alignment(fapl, 0, 8);

      /* If it is not one of these files, return an error */
      } else {
            fprintf(stderr, "Unknown target file!\n");
            return 1;
      }

      return 0;
}

int main (void)
{
      hid_t                   obj, dset, lapl;
      H5L_elink_traverse_t    cb;
      void                    *op_data;
      unsigned                flags;

      /* Create a link access property list */
      lapl = H5Pcreate(H5P_LINK_ACCESS);

      /* Make sure there are no access flags already set on lapl. Of course, there
       * will not be, as we just created lapl; this simply shows how to use
       * H5Pget_elink_acc_flags. */
      H5Pget_elink_acc_flags(lapl, &flags);

      If (!flags)
            /* Set lapl to open external link target files in read only mode.
             * This can be overridden by the callback function. */
            H5Pset_elink_acc_flags(lapl, H5F_ACC_RDONLY);

      /* Make sure there is no callback already set on lapl */
      H5Pget_elink_cb(lapl, &cb, &op_data);

      If (!cb)
```

```
            /* Set the external link traversal callback */
            H5Pset_elink_cb(lapl, elink_callback, NULL);

      /* Open the parent file */
      file = H5Fopen(FILE0, H5F_ACC_RDWR, H5P_DEFAULT);

      /* Open an object from the parent file, which may reside behind one or more
       * external links. */
      obj = H5Oopen(file, OBJ_NAME, lapl);

      …

      return 0;
}
```

Revision History

December 5, 2008: Version 1 circulated for comment within The HDF Group.

January 13, 2008: Version 2 circulated for comment within The HDF Group.

March 3, 2008: Version 3 is published and posted for public comment. Comments should be sent to nfortne2@hdfgroup.org or help@hdfgroup.org