H5P_SET_FILE_IMAGE_CALLBACKS

Expand all Collapse all

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

H5P_SET_FILE_IMAGE_CALLBACKS

Sets the callbacks for working with file images

Motivation: H5P_SET_FILE_IMAGE_CALLBACKS and other elements of HDF5 are used to load an image of an HDF5 file into system memory and open that image as a regular HDF5 file. An application can then use the file without the overhead of disk I/O.

Recommended Reading: This function is part of the file image operations feature set. It is highly recommended to study the guide HDF5 File Image Operations before using this feature set. See the “See Also” section below for links to other elements of HDF5 file image operations.

Procedure:

H5P_SET_FILE_IMAGE_CALLBACKS (fapl_id, callbacks_ptr)

Signature:

```
herr_t H5Pset_file_image_callbacks(
            hid_t fapl_id,
            H5_file_image_callbacks_t *callbacks_ptr
        )
```

Parameters:

hid_t `fapl_id`	IN: File access property list identifier

H5_file_image_callbacks_t *`callbacks_ptr`	IN/OUT: Pointer to an instance of the `H5_file_image_callbacks_t` structure

Description:

H5P_SET_FILE_IMAGE_CALLBACKS sets callback functions for working with file images in memory.

H5P_SET_FILE_IMAGE_CALLBACKS allows an application to control the management of file image buffers through user defined callbacks. These callbacks can be used in the management of file image buffers in property lists and with certain file drivers.

H5P_SET_FILE_IMAGE_CALLBACKS must be used before any file image has been set in the file access property list. Once a file image has been set, the function will fail.

The callback routines set up by H5P_SET_FILE_IMAGE_CALLBACKS are invoked when a new file image buffer is allocated, when an existing file image buffer is copied or resized, or when a file image buffer is released from use.

Some file drivers allow the use of user-defined callback functions for allocating, freeing, and copying the driver's internal buffer, potentially allowing optimizations such as avoiding large `malloc` and `memcpy` operations, or to perform detailed logging.

From the perspective of the HDF5 library, the operations of the `image_malloc`, `image_memcpy`, `image_realloc`, and `image_free` callbacks must be identical to those of the corresponding C standard library calls (`malloc`, `memcpy`, `realloc`, and `free`). While the operations must be identical, the file image callbacks have more parameters. The return values of `image_malloc` and `image_realloc` are identical to the return values of `malloc` and `realloc`. The return values of `image_memcpy` and `image_free` differ from the return values of `memcpy` and `free` in that the return values of `image_memcpy` and `image_free` can also indicate failure.

The callbacks and their parameters, along with a struct and an ENUM required for their use, are described below.

Callback struct and ENUM:

The callback functions set up by H5P_SET_FILE_IMAGE_CALLBACKS use a struct and an ENUM that are defined as follows

The struct **`H5_file_image_callbacks_t`** serves as a container for the callback functions and a pointer to user-supplied data. The struct is defined as follows:

```
typedef struct
    {
        void *(*image_malloc)(size_t size, H5_file_image_op_t file_image_op,
                        void *udata);
        void *(*image_memcpy)(void *dest, const void *src, size_t size,
                        H5_file_image_op_t file_image_op, void *udata);
        void *(*image_realloc)(void *ptr, size_t size,
                        H5_file_image_op_t file_image_op, void *udata);
        herr_t (*image_free)(void *ptr, H5_file_image_op_t file_image_op,
                        void *udata);
        void *(*udata_copy)(void *udata);
        herr_t (*udata_free)(void *udata);
        void *udata;
    } H5_file_image_callbacks_t;
```

Elements of the ENUM **`H5_file_image_op_t`** are used by the callbacks to invoke certain operations on file images. The ENUM is defined as follows:

```
typedef enum
    {
        H5_FILE_IMAGE_OP_PROPERTY_LIST_SET,
        H5_FILE_IMAGE_OP_PROPERTY_LIST_COPY,
        H5_FILE_IMAGE_OP_PROPERTY_LIST_GET,
        H5_FILE_IMAGE_OP_PROPERTY_LIST_CLOSE,
        H5_FILE_IMAGE_OP_FILE_OPEN,
        H5_FILE_IMAGE_OP_FILE_RESIZE,
        H5_FILE_IMAGE_OP_FILE_CLOSE
    } H5_file_image_op_t;
```

The elements of the `H5_file_image_op_t` ENUM are used in the callbacks for the following purposes:

H5_FILE_IMAGE_OP_PROPERTY_LIST_SET	Passed to the `image_malloc` and `image_memcpy` callbacks when a file image buffer is to be copied while being set in a file access property list (FAPL)
H5_FILE_IMAGE_OP_PROPERTY_LIST_COPY	Passed to the `image_malloc` and `image_memcpy` callbacks when a file image buffer is to be copied when a FAPL is copied
H5_FILE_IMAGE_OP_PROPERTY_LIST_GET	Passed to the `image_malloc` and `image_memcpy` callbacks when a file image buffer is to be copied while being retrieved from a FAPL
H5_FILE_IMAGE_OP_PROPERTY_LIST_CLOSE	Passed to the `image_free` callback when a file image buffer is to be released during a FAPL close operation
H5_FILE_IMAGE_OP_FILE_OPEN	Passed to the `image_malloc` and `image_memcpy` callbacks when a file image buffer is to be copied during a file open operation While the file image being opened will typically be copied from a FAPL, this need not always be the case. For example, the core file driver, also known as the memory file driver, takes its initial image from a file.
H5_FILE_IMAGE_OP_FILE_RESIZE	Passed to the `image_realloc` callback when a file driver needs to resize an image buffer
H5_FILE_IMAGE_OP_FILE_CLOSE	Passed to the `image_free` callback when an image buffer is to be released during a file close operation

Callback functions

The **`image_malloc`** callback contains a pointer to a function that must appear to HDF5 to have functionality identical to that of the standard C library `malloc()` call.

Signature in `H5_file_image_callbacks_t`:

```
void *(*image_malloc) ( size_t size, H5_file_image_op_t *file_image_op, void *udata )
```

Parameters:

size_t `size`	IN: Size in bytes of the file image buffer to allocate
H5_file_image_op_t `*file_image_op`	IN: A value from `H5_file_image_op_t` indicating the operation being performed on the file image when this callback is invoked
void `*udata`	IN: Value passed in in the H5P_SET_FILE_IMAGE_CALLBACKS parameter `udata`

The **`image_memcpy`** callback contains a pointer to a function that must appear to HDF5 to have functionality identical to that of the standard C library `memcopy()` call, except that it returns a `NULL` on failure. (The `memcpy` C Library routine is defined to return the `dest` parameter in all cases.)

Setting `image_memcpy` to `NULL` indicates that HDF5 should invoke the standard C library `memcpy()` routine when copying buffers.

Signature in `H5_file_image_callbacks_t`:

void `*(*image_memcpy) (` *void* `*dest`, *const void* `*src`, *size_t* `size`, *H5_file_image_op_t* `*file_image_op`, *void* `*udata )`

Parameters:

void `*dest`	IN: Address of the destination buffer
const void `*src`	IN: Address of the source buffer
size_t `size`	IN: Size in bytes of the file image buffer to copy
H5_file_image_op_t `*file_image_op`	IN: A value from `H5_file_image_op_t` indicating the operation being performed on the file image when this callback is invoked

void *udata	IN: Value passed in in the H5Pset_file_image_callbacks parameter udata

The **image_realloc** callback contains a pointer to a function that must appear to HDF5 to have functionality identical to that of the standard C library realloc() call.

Setting image_realloc to NULL indicates that HDF5 should invoke the standard C library realloc() routine when resizing file image buffers.

Signature in H5_file_image_callbacks_t:

void *(*image_realloc) (*void* *ptr, *size_t* size, *H5_file_image_op_t* *file_image_op, *void* *udata)

Parameters:

void *ptr	IN: Pointer to the buffer being reallocated
size_t size	IN: Desired size in bytes of the file image buffer after reallocation
H5_file_image_op_t *file_image_op	IN: A value from H5_file_image_op_t indicating the operation being performed on the file image when this callback is invoked
void *udata	IN: Value passed in in the H5Pset_file_image_callbacks parameter udata

The **image_free** callback contains a pointer to a function that must appear to HDF5 to have functionality identical to that of the standard C library free() call, except that it will return 0 (SUCCEED) on success and -1 (FAIL) on failure.

Setting image_free to NULL indicates that HDF5 should invoke the standard C library free() routine when releasing file image buffers.

Signature in H5_file_image_callbacks_t:

herr_t (*image_free) (*void* *ptr, *H5_file_image_op_t* *file_image_op, *void* *udata)

Parameters:

void *ptr	IN: Pointer to the buffer being released
H5_file_image_op_t *file_image_op	IN: A value from H5_file_image_op_t indicating the operation being performed on the file image when this callback is invoked
void *udata	IN: Value passed in in the H5Pset_file_image_callbacks parameter udata

The **udata_copy** callback contains a pointer to a function that, from the perspective of HDF5, allocates a buffer of suitable size, copies the contents of the supplied udata into the new buffer, and returns the address of the new buffer. The function returns NULL on failure. This function is necessary if a non-NULL udata parameter is supplied, so that property lists containing the image callbacks can be copied. If the udata parameter below is NULL, then this parameter should be NULL as well.

Signature in H5_file_image_callbacks_t:

void *(*udata_copy) (*void* *udata)

Parameters:

void *udata	IN: Value passed in in the H5Pset_file_image_callbacks parameter udata

The **udata_free** callback contains a pointer to a function that, from the perspective of HDF5, frees a user data block. This function is necessary if a non-NULL udata parameter is supplied so that property lists containing image callbacks can be discarded without a memory leak. If the udata parameter below is NULL, this parameter should be NULL as well.

Signature in H5_file_image_callbacks_t:

herr_t (*udata_free) (*void* *udata)

Parameters:

void *udata	IN: Value passed in in the H5Pset_file_image_callbacks parameter udata

udata, the final field in the H5_file_image_callbacks_t struct, provides a pointer to user-defined data. This pointer will be passed to the image_malloc, image_memcpy, image_realloc, and image_free callbacks. Define udata as NULL if no user-defined data is provided.

Returns:

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

Failure Modes: Due to interactions between this function and H5P_SET_FILE_IMAGE and H5P_GET_FILE_IMAGE, H5P_SET_FILE_IMAGE_CALLBACKS will fail if a file image has already been set in the target file access property list, fapl_id.

Example:

Coming Soon!

See Also:

H5LT_OPEN_FILE_IMAGE

H5F_GET_FILE_IMAGE

H5P_SET_FILE_IMAGE

H5P_GET_FILE_IMAGE

H5P_GET_FILE_IMAGE_CALLBACKS

HDF5 File Image Operations in Advanced Topics in HDF5

Within H5P_SET_FILE_IMAGE_CALLBACKS:

Callback struct H5_file_image_callbacks_t
Callback ENUM H5_file_image_op_t

History:

Release	Change
1.8.9	C function introduced in this release.

--- Last Modified: July 25, 2019 | 02:35 PM