H5F_GET_METADATA_READ_RETRY_INFO

H5F_GET_METADATA_READ_RETRY_INFO

Retrieves the collection of read retries for metadata entries with checksum

Procedure:

H5F_GET_METADATA_READ_RETRY_INFO (file_id, info)

Signature:

```
herr_t H5Fget_metadata_read_retry_info( hid_t file_id, H5F_retry_info_t *info )
```

Parameters:

hid_t file_id	IN: Identifier for a currently opened HDF5 file
H5F_retry_info_t *info	OUT: Struct containing the collection of read retries for metadata entries with checksum

Description:

Motivation:

On a system that is not atomic, the library might possibly read inconsistent metadata with checksum when performing single-writer/multiple-reader (SWMR) operations for an HDF5 file. Upon encountering such situations, the library will try reading the metadata again for a set number of times to attempt to obtain consistent data. The maximum number of read attempts used by the library will be either the value set via H5P_SET_METADATA_READ_ATTEMPTS or the library default value when a value is not set.

When the current number of metadata read attempts used in the library is unable to remedy the reading of inconsistent metadata on a system, the user can assess the information obtained via this routine to derive a different maximum value. The information can also be helpful for debugging purposes to identify potential issues with metadata flush dependencies and SWMR implementation in general.

H5F_GET_METADATA_READ_RETRY_INFO retrieves information regarding the number of read retries for metadata entries with checksum for the file `file_id`. This information is reported in the `H5F_retry_info_t` struct defined in `H5Fpublic.h` as follows:

```
/* The number of metadata entries with checksum */
        #define NUM_METADATA_READ_RETRIES         21

        typedef struct H5F_retry_info_t {
            unsigned nbins;
            uint32_t *retries[H5F_NUM_METADATA_READ_RETRY_TYPES];
        } H5F_retry_info_t;
```

`nbins` is the number of bins for each `retries[i]` of metadata entry `i`. It is calculated based on the current number of read attempts used in the library and logarithmic 10.

If read retries are incurred for a metadata entry `i`, the library will allocate memory for `retries[i] (nbins * sizeof(uint32_t)` and store the collection of retries there. If there are no retries for a metadata entry `i`, `retries[i]` will be NULL. After a call to this routine, users should free each `retries[i]` that is non-NULL, otherwise resource leak will occur.

For the library default read attempts of 100 for SWMR access, `nbins` will be 2 as depicted below:

- `retries[i][0]` is the number of 1 to 9 read retries.
- `retries[i][1]` is the number of 10 to 99 read retries.

For the library default read attempts of 1 for non-SWMR access, `nbins` will be 0 and each `retries[i]` will be NULL.

The following table lists the 21 metadata entries of `retries[]`:

Index for `retries[]`	Metadata entries[*]
0	Object header (version 2)
1	Object header chunk (version 2)
2	B-tree header (version 2)
3	B-tree internal node (version 2)
4	B-tree leaf node (version 2)
5	Fractal heap header
6	Fractal heap direct block (optional checksum)
7	Fractal heap indirect block
8	Free-space header
9	Free-space sections
10	Shared object header message table
11	Shared message record list
12	Extensive array header
13	Extensive array index block
14	Extensive array super block
15	Extensive array data block
16	Extensive array data block page
17	Fixed array header
18	Fixed array data block
19	Fixed array data block page
20	File’s superblock (version 2)

[*] *All entries are of version 0 (zero) unless indicated otherwise.*

Returns:

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

Failure Modes:

When the input identifier is not a file identifier.

When the pointer to the output structure is NULL.

When memory allocation failed for `retries`.

Example:

Example Usage:

Tis example illustrates the case on retrieving the collection of read retries for a file opened with SWMR access.

```
    H5F_retry_info_t info;

    /* Get a copy of file access property list */
    fapl = H5Pcreate(H5P_FILE_ACCESS);

    /* Set to use the latest library format */
    H5Pset_libver_bounds(fapl, H5F_LIBVER_LATEST, H5F_LIBVER_LATEST);

    /* Create a file with the latest library format */
    fid = H5Fcreate(filename, H5F_ACC_TRUNC, H5P_DEFAULT, fapl);

    /* Create groups/datasets etc. in the file */
    :
    :
    :

    /* Close the file */
    H5Fclose(fid);

    /* Open and perform operations via a writer. */
    fidw = H5Fopen(filename, H5F_ACC_RDWR |H5F_ACC_SWMR_WRITE, fapl);
    :
    :
    :
    :
    :
    /* Open and perform operations via a reader */
    fidr = H5Fopen(FILE, H5F_ACC_RDONLY|H5F_ACC_SWMR_READ, fapl);
    :
    :
    :
    :
    :
    /* Retrieve the collection of read retries for the file */
    H5Fget_metadata_read_retry_info(fidr, &info);

    /* Print the collection of read retries */
    for(i = 0; i < NUM_METADATA_READ_RETRIES; i++) {
        if(info. retries[i] != NULL) {
            printf("Read retries for metadata entry %u:\n", i);

            /* info.nbins will be 2 */
            for(j = 0; j < info.nbins; j++)
                /*
                 * Print the following if nonzero:
                 * info.retries[i][0] for # of 1-9 read retries
                 * info.retries[i][1] for # of 10-99 read retries
                 */
        } /* end if */
    } /* end for */

    /* Free the array of retries */
    for(i = 0; i < NUM_ METADATA_READ_RETRIES; i++)
        if(info.retries[i] != NULL)
            free(info.retries[i];
    :
    :
    :
```

History:

Release	Change
1.10.0	C function introduced with this release

--- Last Modified: December 18, 2018 | 03:40 PM