H5F_SET_MDC_CONFIG

H5F_SET_MDC_CONFIG

Attempts to configure metadata cache of target file

Procedure:

H5F_SET_MDC_CONFIG (file_id, config_ptr)

Signature:

```
herr_t H5Fset_mdc_config(hid_t file_id, H5AC_cache_config_t *config_ptr)
```

Parameters:

hid_t `file_id`	IN: Identifier of the target file
H5AC_cache_config_t * `config_ptr`	IN: Pointer to the instance of `H5AC_cache_config_t` containing the desired configuration. The fields in this structure can be placed in several categories: General configuration Increment configuration Decrement configuration Parallel configuration Specific fields are described below.
General configuration fields:	
int `version`	IN: Integer field indicating the the version of the `H5AC_cache_config_t` in use. This field should be set to `H5AC__CURR_CACHE_CONFIG_VERSION` (defined in `H5ACpublic.h`).

hbool_t `rpt_fcn_enabled`	IN: Boolean flag indicating whether the adaptive cache resize report function is enabled. This field should almost always be set to disabled (`0`). Since resize algorithm activity is reported via stdout, it MUST be set to disabled (`0`) on Windows machines. The report function is not supported code, and can be expected to change between versions of the library. Use it at your own risk.
hbool_t `open_trace_File`	IN: Boolean field indicating whether the `trace_file_name` field should be used to open a trace file for the cache. The trace file is a debuging feature that allows the capture of top level metadata cache requests for purposes of debugging and/or optimization. This field should normally be set to `0`, as trace file collection imposes considerable overhead. This field should only be set to `1` when the `trace_file_name` contai ns the full path of the desired trace file, and either there is no open trace file on the cache, or the `close_trace_file` field is also `1`. The trace file feature is unsupported unless used at the direction of The HDF Group. It is intended to allow The HDF Group to collect a trace of cache activity in cases of occult failures and/or poor performance seen in the field, so as to aid in reproduction in the lab. If you use it absent the direction of The HDF Group, you are on your own.
hbool_t `close_trace_file`	IN: Boolean field indicating whether the current trace file (if any) should be closed. See the above comments on the `open_trace_file` field. This field should be set to `0` unless there is an open trace file on the cache that you wish to close. The trace file feature is unsupported unless used at the direction of The HDF Group. It is intended to allow The HDF Group to collect a trace of cache activity in cases of occult failures and/or poor performance seen in the field, so as to aid in reproduction in the lab. If you use it absent the direction of The HDF Group, you are on your own.
char `trace_file_name[]`	IN: Full path of the trace file to be opened if the `open_trace_file` fi eld is set to `1`. In the parallel case, an ascii representation of the mpi rank of the process will be appended to the file name to yield a unique trace file name for each process. The length of the path must not exceed `H5AC__MAX_TRACE_FILE_N AME_LEN` characters. The trace file feature is unsupported unless used at the direction of The HDF Group. It is intended to allow The HDF Group to collect a trace of cache activity in cases of occult failures and/or poor performance seen in the field, so as to aid in reproduction in the lab. If you use it absent the direction of The HDF Group, you are on your own.

hbool_t `evictions_enabled`	IN: A boolean flag indicating whether evictions from the metadata cache are enabled. This flag is initially set to enabled (`1`). In rare circumstances, the raw data throughput requirements may be so high that the user wishes to postpone metadata writes so as to reserve I/O throughput for raw data. The `evictions_enabled` field exists to allow this. However, this is an extreme step, and you have no business doing it unless you have read the User Guide section on metadata caching, and have considered all other options carefully. The `evictions_enabled` field may not be set to disabled (`0`) unless all adaptive cache resizing code is disabled via the `incr_mode`, `flash_incr_mode`, and `decr_mode` fields. When this flag is set to disabled (`0`), the metadata cache will not attempt to evict entries to make space for new entries, and thus will grow without bound. Evictions will be re-enabled when this field is set back to `1`. This should be done as soon as possible.
hbool_t `set_initial_size`	IN: Boolean flag indicating whether the cache should be forced to the user specified initial size.
size_t `initial_size`	IN: If `set_initial_size` is set to `1`, then `initial_size` must contain the desired initial size in bytes. This value must lie in the closed interval `[min_size, max_size]`. (see below)
double `min_clean_fraction`	IN: This field specifies the minimum fraction of the cache that must be kept either clean or empty. The value must lie in the interval [0.0, 1.0]. 0.01 is a good place to start in the serial case. In the parallel case, a larger value is needed -- see Metadata Caching in HDF5 in the collection “Advanced Topics in HDF5.”
size_t `max_size`	IN: Upper bound (in bytes) on the range of values that the adaptive cache resize code can select as the maximum cache size.
size_t `min_size`	IN: Lower bound (in bytes) on the range of values that the adaptive cache resize code can select as the maximum cache size.
long int `epoch_length`	IN: Number of cache accesses between runs of the adaptive cache resize code. 50,000 is a good starting number.
Increment configuration fields:	
enum H5C_cache_incr_mode `incr_mode`	IN: Enumerated value indicating the operational mode of the automatic cache size increase code. At present, only two values are legal: H5C_incr__off: Automatic cache size increase is disabled, and the remaining increment fields are ignored. H5C_incr__threshold: Automatic cache size increase is enabled using the hit rate threshold algorithm.
double `lower_hr_threshold`	IN: Hit rate threshold used by the hit rate threshold cache size increment algorithm. When the hit rate over an epoch is below this threshold and the cache is full, the maximum size of the cache is multiplied by increment (below), and then clipped as necessary to stay within max_size, and possibly max_increment. This field must lie in the interval [0.0, 1.0]. 0.8 or 0.9 is a good starting point.

double `increment`	IN: Factor by which the hit rate threshold cache size increment algorithm multiplies the current maximum cache size to obtain a tentative new cache size. The actual cache size increase will be clipped to satisfy the max_size specified in the general configuration, and possibly max_increment below. The parameter must be greater than or equal to 1.0 -- 2.0 is a reasonable value. If you set it to 1.0, you will effectively disable cache size increases.
hbool_t `apply_max_increment`	IN: Boolean flag indicating whether an upper limit should be applied to the size of cache size increases.
size_t `max_increment`	IN: Maximum number of bytes by which cache size can be increased in a single step -- if applicable.
enum H5C_cache_flash_incr_mode `flash_incr_mode`	IN: Enumerated value indicating the operational mode of the flash cache size increase code. At present, only the following values are legal: H5C_flash_incr__off: Flash cache size increase is disabled. H5C_flash_incr__add_space: Flash cache size increase is enabled using the add space algorithm.
double `flash_threshold`	IN: The factor by which the current maximum cache size is multiplied to obtain the minimum size entry / entry size increase which may trigger a flash cache size increase. At present, this value must lie in the range [0.1, 1.0].
double `flash_multiple`	IN: The factor by which the size of the triggering entry / entry size increase is multiplied to obtain the initial cache size increment. This increment may be reduced to reflect existing free space in the cache and the `max_size` field above. At present, this field must lie in the range [0.1, 10.0].
Decrement configuration fields:	
enum H5C_cache_decr_mode `decr_mode`	IN: Enumerated value indicating the operational mode of the automatic cache size decrease code. At present, the following values are legal: H5C_decr__off: Automatic cache size decrease is disabled. H5C_decr__threshold: Automatic cache size decrease is enabled using the hit rate threshold algorithm. H5C_decr__age_out: Automatic cache size decrease is enabled using the ageout algorithm. H5C_decr__age_out_with_threshold: Automatic cache size decrease is enabled using the ageout with hit rate threshold algorithm
double `upper_hr_threshold`	IN: Hit rate threshold for the hit rate threshold and ageout with hit rate threshold cache size decrement algorithms. When decr_mode is H5C_decr__threshold, and the hit rate over a given epoch exceeds the supplied threshold, the current maximum cache size is multiplied by decrement to obtain a tentative new (and smaller) maximum cache size. When decr_mode is H5C_decr__age_out_with_threshold, there is no attempt to find and evict aged out entries unless the hit rate in the previous epoch exceeded the supplied threshold. This field must lie in the interval [0.0, 1.0]. For H5C_incr__threshold, .9995 or .99995 is a good place to start. For H5C_decr__age_out_with_threshold, .999 might be more useful.

double `decrement`	IN: In the hit rate threshold cache size decrease algorithm, this parameter contains the factor by which the current max cache size is multiplied to produce a tentative new cache size. The actual cache size decrease will be clipped to satisfy the min_size specified in the general configuration, and possibly max_decrement below. The parameter must be be in the interval [0.0, 1.0]. If you set it to 1.0, you will effectively disable cache size decreases. 0.9 is a reasonable starting point.
hbool_t `apply_max_decrement`	IN: Boolean flag indicating whether an upper limit should be applied to the size of cache size decreases.
size_t `max_decrement`	IN: Maximum number of bytes by which the maximum cache size can be decreased in any single step -- if applicable.
int `epochs_before_eviction`	IN: In the ageout based cache size reduction algorithms, this field contains the minimum number of epochs an entry must remain unaccessed in cache before the cache size reduction algorithm tries to evict it. 3 is a reasonable value.
hbool_t `apply_empty_reserve`	IN: Boolean flag indicating whether the ageout based decrement algorithms will maintain a empty reserve when decreasing cache size.
double `empty_reserve`	IN: Empty reserve as a fraction of maximum cache size if applicable. When so directed, the ageout based algorithms will not decrease the maximum cache size unless the empty reserve can be met. The parameter must lie in the interval [0.0, 1.0]. 0.1 or 0.05 is a good place to start.
Parallel configuration field:	
int `dirty_bytes_threshold`	IN: Threshold number of bytes of dirty metadata generation for triggering synchronizations of the metadata caches serving the target file in the parallel case. Synchronization occurs whenever the number of bytes of dirty metadata created since the last synchronization exceeds this limit. This field only applies to the parallel case. While it is ignored elsewhere, it can still draw a value out of bounds error. It must be consistant across all caches on any given file. By default, this field is set to 256 KB. It shouldn't be more than half the current maximum cache size times the minimum clean fraction.

Description:

H5F_SET_MDC_CONFIG attempts to configure the file's metadata cache according configuration supplied in `*config_ptr`.

See the overview of the metadata cache in the special topics section of the user manual for details on what is being configured. If you haven't read and understood that documentation, you really shouldn't be using this API call.

Returns:

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

Example:

History:

Release	Change

1.8.0	Function introduced in this release.

--- Last Modified: July 22, 2020 | 03:04 PM