# HDF5 Performance Framework C API

This document provides a description of C APIs that are essential for creating custom benchmark programs.

HPF C API is composed of four parts: general, command line, storage and utility.

## 1 General API

**Name:**

- H5Perf_init

**Signature:**

- int H5Perf_init()

**Purpose:**

- Initializes the C API by initializing the required data structures like look-up tables to manage handlers.

**Parameters:**

- None

**Returns**:

- (-1) if fails, zero otherwise

---

**Name:**

- H5Perf_end

**Signature:**

- int H5Perf_end()

**Purpose:**

- Closes the C API library. This must be called before a program terminates.

**Parameters:**

- None

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

## 2 Command Line API

These APIs are obsolete. Feel free to ignore them.

**Name:**

- H5Perf_createCommandLine

**Signature:**

- long int H5Perf_createCommandLine(const char* message, const char* version)

**Purpose:**

- Creates a command line object and returns its handle for user's future calls. The command line object represents the specifications of all command line arguments. The *h* flag is provided by default for help and usage message.

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_addCharArgument

**Signature:**

- int H5Perf_addCharArgument(long int cmd_handle, const char* flag, const char* name, const char* desc, int req, char value)

**Purpose:**

- Adds an argument to the previously created command line Object referenced by cmd_handle. The command line object must be already created using H5Perf_createSetting() function. The type of the argument is character.

**Parameters:**

- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *name*: The long key for the argument (like --measure)
- *desc* The argument description printed when using the default help option by user
- *req* Non-zero for mandatory arguments
- *value* Default value when nothing is provided

**Returns**:

- (-1) if fails, zero otherwise

---

**Name:**

- H5Perf_addIntArgument

**Signature:**

- int H5Perf_addIntArgument(long int cmd_handle, const char* flag, const char* name, const char* desc, int req, int value)

**Purpose:**

- Adds an argument to the previously created command line Object referenced by cmd_handle. The command line object must be created before using H5Perf_createSetting() function. The type of the argument is integer.

**Parameters**:

- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *name* The long key for the argument (like --measure)
- *desc* The argument description printed when using the default help option by user
- *req* Non-zero for mandatory arguments
- *value* Default value when nothing is provided

**Returns**:

- (-1) if fails, zero otherwise

---

**Name:**

- H5Perf_addStringArgument

**Signature:**

- int H5Perf_addStringArgument(long int cmd_handle, const char* flag, const char* name, const char* desc, int req, char* value)

**Purpose:**

- Adds an argument to the previously created command line Object referenced by cmd_handle. The command line object must be created before using H5Perf_createSetting() function. The type of the argument is string.

**Parameters**:

- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *name* The long key for the argument (like --measure)

- *desc* The argument description printed when using the default h option by user
- *req* Non-zero for mandatory arguments
- *value* Default value when nothing is provided

**Returns**:
- (-1) if fails, zero otherwise

---

**Name:**
- H5Perf_parse

**Signature**
- int H5Perf_parse(long int cmd_handle,int argc, char** argv)

**Purpose:**
- Parses an instance of command line complying with the defined command line object and provides the values for future use by user through argument getter functions.

**Parameters**:
- *cmd_handle* The command line object handle
- *argc* The standard main function argc parameter.
- *argv* The standard main function argv parameter.

**Returns**:
- (-1) if fails, zero otherwise

---

**Name:**
- H5Perf_getCharValue

**Signature:**
- int H5Perf_getCharValue(long int cmd_handle, const char* flag, char* value)

**Purpose:**
- Gets the parsed command line character argument

**Parameters**:
- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *value* Argument value

**Returns**:
- (-1) if fails, zero otherwise

---

**Name:**

- H5Perf_getIntValue

**Signature:**

- int H5Perf_getIntValue(long int cmd_handle, const char* flag, int* value)

**Purpose:**

- Gets the parsed command line integer argument

**Parameters**:

- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *value* Argument value

**Returns**:

- (-1) if fails, zero otherwise

---

**Name:**

- H5Perf_getStringValue

**Signature:**

- int H5Perf_getStringValue(long int cmd_handle, const char* flag, char* value)

**Purpose:**

- Gets the parsed command line string argument

**Parameters**:

- *cmd_handle* The command line object handle
- *flag* The short key for the argument ( like -m)
- *value* Argument value

**Returns**:

- (-1) if fails, zero otherwise

## 3 Storage API

**Name:**

- long int H5Perf_createSetting()

**Purpose:**

- Creates a setting object and returns its handle for user's future calls.

**Parameters**:
- None

**Returns**:
- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**
- H5Perf_addSetting

**Signature:**
- int H5Perf_addSetting(long int setting_handle, char* name, char* value)

**Purpose:**
- Adds an entry to a previously created setting object. The object must be created before, using the H5Perf_createSetting() function.

**Parameters**:
- *setting_handle* The setting object handle
- *name* Name of the environmetal setting
- *value* Value of the environmetal setting

**Returns**:
- (-1) if fails, zero otherwise

---

**Name:**
- H5Perf_setSetting

**Signature:**
- int H5Perf_setSetting(long int setting_handle, char* name, char* value)

**Purpose:**
- Sets an entry of the previously created Setting Object referenced by handle. The object must be already created using H5Perf_createSetting() function, and the name must be added to the created object before using H5Perf_addSetting() function.That is you need to call H5Perf_createSetting() first, H5Perf_addSetting() second, and H5Perf_setSetting() at the end.

**Parameters**:
- *setting_handle* The setting object handle

- *name* Name of the environmental setting
- *value* Value of the environmental setting

**Returns**:
- (-1) if fails, zero otherwise

---

**Name:**
- H5Perf_createRoutine

**Signature:**
- long int H5Perf_createRoutine()

**Parameters**:
- None

**Purpose:**
- Creates a Test Routine object and returns its handle for user's future calls.

**Returns**:
- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**
- H5Perf_setRoutine

**Signature:**
- int H5Perf_setRoutine(long int routine_handle,char* name, char* description, char* version, long int setting_handle)

**Purpose:**
- Sets a previously created Test Routine object referenced by routine_handle. The object must be created before using H5Perf_createRoutine() function.

**Parameters**:
- *routine_handle* The Test Routine object handle
- *name* Name of the Test Routine
- *description* Description of the Test Routine
- *version* Version of the Test Routine
- *setting_handle* handler for Test Routine setting

**Returns**:
- (-1) if fails, greater or equal to zero otherwise

---

**Name:**

- H5Perf_addAction

**Signature:**

- int H5Perf_addAction(long int routine_handle, char* name, char* description, long int setting_handle)

**Purpose:**

- Adds Test Action information to a previously created Test Routine object referenced by routine_handle. The object must be created before using H5Perf_createRoutine() and H5Perf_setRoutine() functions.

**Parameters**:

- routine_handle The TestRoutine object handle
- name Name of the Test action
- description Description of the Test Action
- settings_handle Handler for Test Action setting

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

---

**Name:**

- H5Perf_addInstance

**Signature:**

- int H5Perf_addInstance(long int routine_handle,char* action_name,char* datasetName, char* datasetDesc,const char* host, unsigned int year,unsigned int month,unsigned int day,unsigned int hour,unsigned int minute, unsigned int second, char* libVersion, double result, long int setting_handle)

**Purpose:**

- Adds Test instance information to a previously created Test Routine object referenced by routine_handle. The object must be created before using H5Perf_createRoutine() and H5Perf_setRoutine() functions. Also it must already have one action with name action_name set by H5Perf_addAction function.

**Parameters**:

- *routine_handle* The TestRoutine object handle
- *action_name* Name of the Test action
- *datasetName* Name of the Test Instance dataset
- *datasetDesc* Description of the Test Instance dataset
- *host* The host name this instance is running on
- *year,..* The running date
- *libVersion* HDF5 library version used
- *result* The rest result

- *setting_handle* Handler for Test Instance setting

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

---

**Name:**

- H5Perf_createOneInstanceRoutine

**Signature:**

- long int H5Perf_createOneInstanceRoutine(char* routineName, char* datasetName, char* datasetDesc,const char* host, unsigned int year, unsigned int month, unsigned int day, unsigned int hour, unsigned int minute, unsigned int second, char* libVersion, double result, long int setting_handle)

**Purpose:**

- Creates a Test Routine object with just one Test Instance and returns its handle. The handle could be used for user's future call on this Test Routine object by H5Perf_setOneInstanceRoutineResult function. Using this function, user could ignore the usual order of creating/adding Test objects and directly creates a Test Instance object.
- This API is obsolete. Feel free to ignore this API.

**Parameters**:

- *routine_name* Name of the Test Routine
- *datasetName* Name of the Test Instance dataset
- *datasetDesc* Description of the Test Instance dataset
- *host* The host name this instance is running on
- *year,..* The running date
- *libVersion* HDF5 library version used
- *setting_handle* Handler for Test Instance setting

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_createFileHandle

**Signature:**

- long int H5Perf_createFileHandle(char* parent,char* name, int append)

**Purpose:**

- Creates a File handle and returns it for user's future calls. This method also opens the file.

**Parameters**:

- *parent* The file parent directory
- *name* The file name
- *append* Non-zero to append to write the end of previously created file.

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_createMySQLHandle

**Signature:**

- long int H5Perf_createMySQLHandle(char* server,char* dbname, char* uid, char* passwd, int port)

**Purpose:**

- Creates a handle to MySQL database and saves it for user's future calls. This method also opens a connection to the DBMS.

**Parameters**:

- *server* The server host address
- *dbname* The database name
- *uid* User name
- *passwd* Password
- *port* Port number of the DBMS on the server host

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_close

**Signature:**

- int H5Perf_close(long int handle)

**Purpose:**

- Closes the object referenced by the handle and frees its dependent resources. If the object is a storage system, the connection will be closed (file handle or database connection).

**Parameters**:
- *handle* The handle to the previously created object

**Returns**:
- (-1) if fails, greater or equal to zero otherwise

---

**Name:**
- H5Perf_find_routine

**Signature:**
- long int H5Perf_find_routine(long int randomStorage_handle, char* routine_name)

**Purpose:**
- Finds the routine object with name *routine_name* as well as its dependent objects from the storage that must be random access storage like MySQL.

**Parameters**:
- *randomStorage_hanlde* The handle to the previously created random access storage object (MySQL)
- *routine_name* The name of the routine object

**Returns**:
- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**
- H5Perf_find_action

**Signature:**
- long int H5Perf_find_action(long int routine_handle, char* action_name)

**Purpose:**
- Finds the action object with name *action_name* from a storage.

**Parameters**:
- *routine_handle* The handle to the previously created routine object
- *action_name* The name of the action object

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_find_instance

**Signature:**

- long int H5Perf_find_instance(long int action_handle, char* instance_name)

**Purpose:**

- Finds the instance object with name *instnace_name* from a storage.

**Parameters**:

- *action_hanlde* The handle to the previously created action object
- *instance_name* The name of the instance object

**Returns**:

- (-1) if fails, a valid handle otherwise (greater or equal to zero)

---

**Name:**

- H5Perf_write

**Signature:**

- int H5Perf_write (long int storage_handle, long int routine_handle)

**Purpose:**

- Stores the routine object as well as its dependent objects information referenced by routine_handle to the storage that could be file or MySQL DBMS referenced by *randomStorage_handle*.

**Parameters**:

- *randomStorage_hanlde* The handle to the previously created storage object (File/ MySQL)
- *routine_handle* The handle to the routine object

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

---

**Name:**

- H5Perf_update

**Signature:**

- int H5Perf_update (long int randomStorage_handle, long int routine_handle)

**Purpose:**

- Updates the routine object as well as its dependent objects information referenced by routine_handle stored in a random access storage like MySQL DBMS referenced by *randomStorage_handle*.

**Parameters**:

- *randomStorage_hanlde* The handle to the previously created random access storage object (MySQL)
- *routine_handle* The handle to the routine object

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

---

**Name:**

- H5Perf_remove

**Signature:**

- int H5Perf_remove (long int randomStorage_handle, char* routine_name)

**Purpose:**

- Removes the routine object as well as its dependent objects information with name *routine_name* stored in a random access storage like MySQL DBMS referenced by *randomStorage_handle*.

**Parameters**:

- *randomStorage_hanlde* The handle to the previously created random access storage object (MySQL)
- *routine_handle* The handle to the routine object

**Returns**:

- (-1) if fails, greater or equal to zero otherwise

## 4 Utility API

**Name:**

- H5Perf_startTimer

**Signature:**

- void H5Perf_startTimer(struct timeval* timeval_start)

**Purpose**:
- Starts a timer to measure up a time interval

**Parameters**:
- *timeval_start* Standard timeval struct according to sys/time.h

---

**Name:**
- H5Perf_startUsageTimer

**Signature:**
- void H5Perf_startUsageTimer()

**Purpose**:
- Starts a timer based on getrusage() to measure up a time interval

---

**Name:**
- H5Perf_endTimer

**Signature:**
- double H5Perf_endTimer(struct timeval start)

**Purpose:**
- Stops the timer to measure up a time interval

**Parameters**:
- *timeval_start* Standard *timeval struct* according to *sys/time.h* passed to the H5Perf_startTimer before.

**Returns**:
- The time interval value

---

**Name:**
- H5Perf_endUsageTimer

**Signature:**

- void H5Perf_endUsageTimer()

**Purpose:**
- Stops the getrusage()-based timer to measure up a time interval

---

**Name:**
- H5Perf_getUserTime

**Signature:**
- double H5Perf_getUserTime()

**Purpose:**
- gets the elapsed user time between H5Perf_startUsageTimer() and H5Perf_endUsageTimer() calls.

**Returns**:
- The user time interval value
- This measurement is less susceptible to system load.

---

**Name:**
- H5Perf_getSystemTime

**Signature:**
- double H5Perf_getSystemTime()

**Purpose:**
- gets the elapsed system time between H5Perf_startUsageTimer() and H5Perf_endUsageTimer() calls.
- This measurement varies a lot depending on system load.

**Returns**:
- The system time interval value

---

**Name:**
- H5Perf_getRandom

**Signature:**

- double H5Perf_getRandom(long int limit)

**Purpose:**

- Generates a random number in the range of [0,limit)

**Parameters**:

- *limit* The excluded maximum value of generated random number.

**Returns**:

- A random number