HDF5 API Reference

Release 0.1

The HDF Group

November 27, 2014

Contents

CHAPTER 1

HDF5 Image (H5IM)

The HDF5 Image API defines a standard storage for HDF5 datasets that are indented to be interpreted as images. The specification for this API is presented in another document: HDF5 Image and Palette Specification. This version of the API is primarily concerned with two dimensional raster data similar to HDF4 Raster Images. The HDF5 image API uses the Lite HDF5 API.

The following functions are part of the HDF5 Image API version 1.0.

Programming Hints:

To use any of these functions or subroutines, you must first include the relevant include file (C) or module (Fortran) in your application.

The following line includes the HDF5 Image package, `H5IM`, in C applications:

```
#include "hdf5_hl.h"
```

This line includes the `H5IM` module in Fortran applications:

```
use h5im
```

1.1 Image Functions

1.1.1 H5IMget_image_info

herr_t **H5IMget_image_info**(hid_t *loc_id*, const char **dset_name*, hsize_t **width*, hsize_t **height*, hsize_t **planes*, char **interlace*, hssize_t **npals*)

`H5IMget_image_info` gets information about an image named `dset_name` attached to the file or group specified by the identifier `loc_id`.

Synopsis Gets information about an image dataset (dimensions, interlace mode and number of associated palettes).

Parameters

- **loc_id** – IN: Identifier of the file or group in which the dataset is located.
- **dset_name** – IN: The name of the dataset.
- **width** – OUT: The width of the image.
- **height** – OUT: The height of the image.
- **planes** – OUT: The number of color planes of the image.

- **interlace** – OUT: The interlace mode of the image.
- **npals** – OUT: The number of palettes associated to the image.

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

Fortran90 Interface: `h5imget_image_info_f`

```
subroutine h5imget_image_info_f(loc_id, dset_name, width, height, planes, &
                                interlace, npals, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id            ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name       ! name of the dataset
  integer(HSIZE_T), intent(INOUT) :: width        ! width of image
  integer(HSIZE_T), intent(INOUT) :: height       ! height of image
  integer(HSIZE_T), intent(INOUT) :: planes       ! color planes
  integer(HSIZE_T), intent(INOUT) :: npals        ! palettes
  character(LEN=*), intent(INOUT) :: interlace    ! interlace
  integer :: errcode                              ! error code
end subroutine h5imget_image_info_f
```

1.1.2 H5IMis_image

herr_t **H5IMis_image** (hid_t *loc_id*, const char **dset_name*)

`H5IMis_image` inquires if a dataset named `dset_name`, attached to the file or group specified by the identifier `loc_id`, is an image based on the HDF5 Image and Palette Specification.

Synopsis Inquires if a dataset is an image.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the dataset is located.
- **dset_name** – IN: The name of the dataset.

Returns Returns true, false, fail.

Fortran90 Interface: `h5imis_image_f`

```
integer function h5imis_image_f(loc_id, dset_name)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  integer :: errcode                            ! error code
end function h5imis_image_f
```

1.1.3 H5IMmake_image_8bit

herr_t **H5IMmake_image_8bit** (hid_t *loc_id*, const char **dset_name*, hsize_t *width*, hsize_t *height*, const unsigned char **buffer*)

`H5IMmake_image_8bit` creates and writes a dataset named `dset_name` attached to the file or group specified by the identifier `loc_id`. Attributes conforming to the HDF5 Image and Palette specification for an indexed image are attached to the dataset, thus identifying it as an image. The image data is of the type `H5T_NATIVE_UCHAR`. An indexed image is an image in which each each pixel information storage is an index to a table palette.

Synopsis Creates and writes an image.

Parameters

- **loc_id** – IN: Identifier of the file or group to create the dataset within.
- **dset_name** – IN: The name of the dataset to create.
- **width** – IN: The width of the image.
- **height** – IN: The height of the image.
- **buffer** – IN: Buffer with data to be written to the dataset.

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

Fortran90 Interface: `h5immake_image_8bit_f`

```
subroutine h5immake_image_8bit_f(loc_id, dset_name, width, height, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  integer(HSIZE_T), intent(IN) :: width          ! width of image
  integer(HSIZE_T), intent(IN) :: height         ! height of image
  integer*1, intent(IN), dimension(*) :: buf     ! 1 byte integer data buffer
  integer :: errcode                             ! error code
end subroutine h5immake_image_8bit_f
```

1.1.4 H5IMmake_image_24bit

herr_t **H5IMmake_image_24bit** (hid_t *loc_id*, const char **dset_name*, hsize_t *width*, hsize_t *height*, const char **interlace*, const unsigned char **buffer*)

`H5IMmake_image_24bit` creates and writes a dataset named `dset_name` attached to the file or group specified by the identifier `loc_id`. This function defines a true color image conforming to the HDF5 Image and Palette specification. The function assumes that the image data is of the type `H5T_NATIVE_UCHAR`.

A true color image is an image where the pixel storage contains several color planes. In a 24 bit RGB color model, these planes are red, green and blue. In a true color image the stream of bytes can be stored in several different ways, thus defining the interlace (or interleaving) mode. The 2 most used types of interlace mode are interlace by pixel and interlace by plane. In the 24 bit RGB color model example, interlace by plane means all the red components for the entire dataset are stored first, followed by all the green components, and then by all the blue components. Interlace by pixel in this example means that for each pixel the sequence red, green, blue is defined. In this function, the interlace mode is defined in the parameter `interlace`, a string that can have the values `INTERLACE_PIXEL` or `INTERLACE_PLANE`.

Synopsis Creates and writes a true color image.

Parameters

- **loc_id** – IN: Identifier of the file or group to create the dataset within.
- **dset_name** – IN: The name of the dataset to create.
- **width** – IN: The width of the image.
- **height** – IN: The height of the image.
- **interlace** – IN: String defining the interlace mode.
- **buffer** – IN: Buffer with data to be written to the dataset.

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

Fortran90 Interface: `h5immake_image_24bit_f`

```
subroutine h5immake_image_24bit_f(loc_id, dset_name, width, height, il, &
                                  buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  integer(HSIZE_T), intent(IN) :: width         ! width of image
  integer(HSIZE_T), intent(IN) :: height        ! height of image
  character(LEN=*), intent(IN) :: il            ! interlace
  integer*1, intent(IN), dimension(*) :: buf    ! 1 byte integer data buffer
  integer :: errcode                            ! error code
end subroutine h5immake_image_24bit_f
```

1.1.5 H5IMread_image

herr_t **H5IMread_image** (hid_t *loc_id*, const char **dset_name*, unsigned char **buffer*)

H5IMread_image reads a dataset named dset_name attached to the file or group specified by the identifier loc_id.

Synopsis Reads image data from disk.

Parameters

- **loc_id** – IN: Identifier of the file or group to read the dataset from.
- **dset_name** – IN: The name of the dataset.
- **buffer** – OUT: Buffer with data to store the image.

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

Fortran90 Interface: h5imread_image_f

```
subroutine h5imread_image_f(loc_id, dset_name, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id             ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name        ! name of the dataset
  integer*1, intent(INOUT), dimension(*) :: buf    ! 1 byte integer data buffer
  integer :: errcode                               ! error code
end subroutine h5imread_image_f
```

1.2 Palette Functions

1.2.1 H5IMget_npalettes

herr_t **H5IMget_npalettes** (hid_t *loc_id*, const char **image_name*, hssize_t **npals*)

H5IMget_npalettes gets the number of palettes associated to an image specified by image_name.

Synopsis Gets the number of palettes associated to an image.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the imagedataset is located.
- **image_name** – IN: The name of the image dataset.
- **npals** – OUT: The number of palettes.

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

Fortran90 Interface: h5imget_npalettes_f

```
subroutine h5imget_npalettes_f(loc_id, dset_name, npals, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  integer(HSIZE_T), intent(INOUT) :: npals      ! palettes
  integer :: errcode                            ! error code
end subroutine h5imget_npalettes_f
```

1.2.2 H5IMget_palette

herr_t **H5IMget_palette**(hid_t *loc_id*, const char **image_name*, int *pal_number*, unsigned char **pal_data*)

H5IMget_palette gets the palette dataset identified by pal_number (a zero based index) associated to an image specified by image_name.

Synopsis Gets the palette dataset.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the image dataset is located.
- **image_name** – IN: The name of the image dataset.
- **pal_number** – IN: The zero based index that identifies the palette.
- **pal_data** – OUT: The palette dataset.

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

Fortran90 Interface: h5imget_palette_f

```
subroutine h5imget_palette_f(loc_id, dset_name, pal_number, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id            ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name       ! name of the dataset
  integer, intent(IN) :: pal_number               ! palette number
  integer*1, intent(INOUT), dimension(*) :: buf   ! 1 byte integer data buffer
  integer :: errcode                              ! error code
end subroutine h5imget_palette_f
```

1.2.3 H5IMget_palette_info

herr_t **H5IMget_palette_info**(hid_t *loc_id*, const char **image_name*, int *pal_number*, hsize_t **pal_dims*)

H5IMget_palette_info gets the dimensions of the palette dataset identified by pal_number (a zero based index) associated to an image specified by image_name.

Synopsis Gets information about a palette dataset (dimensions).

Parameters

- **loc_id** – IN: Identifier of the file or group in which the image dataset is located.
- **image_name** – IN: The name of the image dataset.
- **pal_number** – IN: The zero based index that identifies the palette.
- **pal_dims** – OUT: The dimensions of the palette dataset.

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

Fortran90 Interface: `h5imget_palette_info_f`

```
subroutine h5imget_palette_info_f(loc_id, dset_name, pal_number, dims, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id            ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name       ! name of the dataset
  integer, intent(IN) :: pal_number               ! palette number
  integer(HSIZE_T), dimension(*), intent(INOUT) :: dims
                                                  ! dimensions
  integer :: errcode                              ! error code
end subroutine h5imget_palette_info_f
```

1.2.4 H5IMis_palette

herr_t **H5IMis_palette** (hid_t *loc_id*, const char **dset_name*)

`H5IMis_palette` inquires if a dataset named `dset_name`, attached to the file or group specified by the identifier `loc_id`, is a palette based on the HDF5 Image and Palette Specification.

Synopsis Inquires if a dataset is a palette.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the image dataset is located.
- **dataset_name** – IN: The name of the dataset.

Returns Returns true, false, fail.

Fortran90 Interface: `h5imis_palette_f`

```
integer function h5imis_palette_f(loc_id, dset_name)
  implicit none
  integer(HID_T), intent(IN) :: loc_id         ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name    ! name of the dataset
  integer :: errcode                           ! error code
end function h5imis_palette_f
```

1.2.5 H5IMlink_palette

herr_t **H5IMlink_palette** (hid_t *loc_id*, const char **image_name*, const char **pal_name*)

`H5IMlink_palette` attaches a palette named `pal_name` to an image specified by `image_name`. The image dataset may or not already have an attached palette. If it has, the array of palette references is extended to hold the reference to the new palette.

Synopsis Attaches a palette to an image.

Parameters

- **loc_id** – IN: Identifier of the file or group.
- **image_name** – IN: The name of the dataset to attach the palette to.
- **pal_name** – IN: The name of the palette.

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

Fortran90 Interface: `h5imlink_palette_f`

```
subroutine h5imlink_palette_f(loc_id, dset_name, pal_name, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  character(LEN=*), intent(IN) :: pal_name      ! palette name
  integer :: errcode                            ! error code
end subroutine h5imlink_palette_f
```

1.2.6 H5IMmake_palette

herr_t **H5IMmake_palette**(hid_t *loc_id*, const char **pal_name*, const hsize_t **pal_dims*, const unsigned char **pal_data*)

`H5IMmake_palette` creates and writes a dataset named `pal_name`. Attributes conforming to the HDF5 Image and Palette specification are attached to the dataset, thus identifying it as a palette. The palette data is of the type `H5T_NATIVE_UCHAR`.

Synopsis Creates and writes a palette.

Parameters

- **loc_id** – IN: Identifier of the file or group to create the dataset within.
- **pal_name** – IN: The name of the palette.
- **pal_dims** – IN: An array of the size of the palette dimensions.
- **pal_data** – IN: Buffer with data to be written to the dataset.

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

Fortran90 Interface: `h5immake_palette_f`

```
subroutine h5immake_palette_f(loc_id, dset_name, pal_dims, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  integer(HSIZE_T), intent(IN), dimension(*) :: pal_dims
                                                ! dimensions
  integer*1, intent(IN), dimension(*) :: buf    ! 1 byte integer data buffer
  integer :: errcode                            ! error code
end subroutine h5immake_palette_f
```

1.2.7 H5IMunlink_palette

herr_t **H5IMunlink_palette**(hid_t *loc_id*, const char **image_name*, const char **pal_name*)

`H5IMunlink_palette` detaches a palette from an image specified by `image_name`.

Synopsis Detaches a palette from an image.

Parameters

- **loc_id** – IN: Identifier of the file or group.
- **image_name** – IN: The name of the image dataset.
- **pal_name** – IN: The name of the palette.

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

Fortran90 Interface: `h5imunlink_palette_f`

```
subroutine h5imunlink_palette_f(loc_id, dset_name, pal_name, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  character(LEN=*), intent(IN) :: pal_name      ! palette name
  integer :: errcode                            ! error code
end subroutine h5imunlink_palette_f
```

CHAPTER 2

HDF5 Table (H5TB)

The HDF5 Table API defines a standard storage for HDF5 datasets that are indented to be interpreted as tables. A table is defined as a collection of records whose values are stored in fixed-length fields. All records have the same structure and all values in each field have the same data type.

The following functions are part of the HDF5 Table API.

Programming Hints:

To use any of these functions or subroutines, you must first include the relevant include file (C) or module (Fortran) in your application.

The following line includes the HDF5 Table package, `H5TB`, in C applications:

```
#include "hdf5_hl.h"
```

This line includes the `H5TB` module in Fortran applications:

```
use h5tb
```

2.1 Creation

2.1.1 H5TBmake_table

herr_t **H5TBmake_table**(const char **table_title*, hid_t *loc_id*, const char **dset_name*, hsize_t *nfields*, const hsize_t *nrecords*, size_t *type_size*, const char *field_names [ ], const size_t **field_offset*, const hid_t **field_types*, hsize_t *chunk_size*, void **fill_data*, int *compress*, const void **data*)

`H5TBmake_table creates` and writes a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

Synopsis Creates and writes a table.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the dataset is located.
- **table_title** – IN: The title of the table.
- **loc_id** – IN: Identifier of the file or group to create the table within.
- **table_name** – IN: The name of the dataset to create.
- **nfields** – IN: The number of fields.

- **nrecords** – IN: The number of records.
- **type_size** – IN: The size in bytes of the structure associated with the table. This value is obtained with `sizeof`.
- **field_names** – IN: An array containing the names of the fields.
- **field_offset** – IN: An array containing the offsets of the fields.
- **field_types** – IN: An array containing the type of the fields.
- **chunk_size** – IN: The chunk size.
- **fill_data** – IN: Fill values data.
- **compress** – IN: Flag that turns compression on or off.
- **data** – IN: Buffer with data to be written to the table.

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

Fortran90 Interface: `h5tbmake_table_f`

Note: `h5tbmake_table_f` only creates the table, it does not write data to it.

```
subroutine h5tbmake_table_f(table_title, loc_id, dset_name, nfields, &
                            nrecords, type_size, field_names, field_offset, &
                            field_types, chunk_size, compress, errcode)
  implicit none
  character(LEN=*), intent(IN) :: table_title     ! name of the table
  integer(HID_T), intent(IN) :: loc_id            ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name       ! name of the dataset
  integer(HSIZE_T), intent(IN) :: nfields         ! fields
  integer(HSIZE_T), intent(IN) :: nrecords        ! records
  integer(SIZE_T), intent(IN) :: type_size        ! type size
  character(LEN=*), dimension(nfields), intent(IN) :: field_names
                                                  ! field names
  integer(SIZE_T), dimension(nfields), intent(IN) :: field_offset
                                                  ! field offset
  integer(HID_T), dimension(nfields), intent(IN) :: field_types
                                                  ! field types
  integer(HSIZE_T), intent(IN) :: chunk_size      ! chunk size
  integer, intent(IN) :: compress                 ! compress
  integer :: errcode                              ! error code
end subroutine h5tbmake_table_f
```

2.2 Storage

2.2.1 H5TBappend_records

herr_t **H5TBappend_records** (hid_t *loc_id*, const char **dset_name*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **field_sizes*, const void **data*)

`H5TBappend_records` adds records to the end of the table named `dset_name` attached to the object specified by the identifier `loc_id`. The dataset is extended to hold the new records.

Synopsis Adds records to the end of the table.

Parameters

- **loc_id** – IN: Identifier of the file or group where the table is located.

- **dset_name** – IN: The name of the dataset to overwrite.
- **nrecords** – IN: The number of records to append.
- **type_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **field_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET` macro.
- **field_sizes** – IN: An array containing the sizes of the fields.
- **data** – IN: Buffer with data.

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

2.2.2 H5TBwrite_records

herr_t **H5TBwrite_records**(hid_t *loc_id*, const char **table_name*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **field_sizes*, const void **data*)

`H5TBwrite_records` overwrites records starting at the zero index position start of the table named `table_name` attached to the object specified by the identifier `loc_id`.

Synopsis Overwrites records.

Parameters

- **loc_id** – IN: Identifier of the file or group where the table is located.
- **table_name** – IN: The name of the dataset to overwrite.
- **start** – IN: The zero index record to start writing.
- **nrecords** – IN: The number of records to write.
- **type_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **field_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET` macro.
- **field_sizes** – IN: An array containing the sizes of the fields.
- **data** – IN: Buffer with data.

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

2.2.3 H5TBwrite_fields_name

herr_t **H5TBwrite_fields_name**(hid_t *loc_id*, const char **table_name*, const char **field_names*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **field_sizes*, const void **data*)

`H5TBwrite_fields_name` overwrites one or several fields contained in the buffer `field_names` from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

Synopsis Overwrites fields.

Parameters

- **loc_id** – IN: Identifier of the file or group where the table is located.
- **table_name** – IN: The name of the dataset to overwrite.
- **field_names** – IN: The names of the fields to write.

- **start** – IN: The zero based index record to start writing.
- **nrecords** – IN: The number of records to write.
- **type_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **field_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET`macro.
- **field_sizes** – IN: An array containing the sizes of the fields.
- **data** – IN: Buffer with data.

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

Fortran90 Interface: `h5tbwrite_field_name_f`

```
subroutine h5tbwrite_field_name_f(loc_id, dset_name, field_name, start, &
                                  nrecords, type_size, buf, errcode)

  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  character(LEN=*), intent(IN) :: field_name     ! name of the field
  integer(HSIZE_T), intent(IN) :: start          ! start record
  integer(HSIZE_T), intent(IN) :: nrecords       ! records
  integer(SIZE_T), intent(IN) :: type_size       ! type size
  <TYPE>, intent(IN), dimension(*) :: buf        ! data buffer
  integer :: errcode                             ! error code

  end subroutine h5tbwrite_field_name_f
```

2.2.4 H5TBwrite_fields_index

herr_t **H5TBwrite_fields_index**(hid_t *loc_id*, const char **table_name*, int *nfields*, const int **field_index*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t* *field_sizes*, const void **data*)

`H5TBwrite_fields_index` overwrites one or several fields contained in the buffer `field_index` from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

Synopsis Overwrites fields.

Parameters

- **loc_id** – IN: Identifier of the file or group where the table is located.
- **table_name** – IN: The name of the dataset to overwrite.
- **nfields** – IN: The number of fields to overwrite. This parameter is also the size of the `field_index` array.
- **field_index** – IN: The indexes of the fields to write.
- **start** – IN: The zero based index record to start writing.
- **nrecords** – IN: The number of records to write.
- **type_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **field_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET` macro.
- **field_sizes** – IN: An array containing the sizes of the fields.

- **data** – IN: Buffer with data.

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

Fortran90 Interface: `h5tbwrite_field_index_f`

```
subroutine h5tbwrite_field_index_f(loc_id, dset_name, field_index, start, &
                                   nrecords, type_size, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  integer, intent(IN) :: field_index             ! index
  integer(HSIZE_T), intent(IN) :: start          ! start record
  integer(HSIZE_T), intent(IN) :: nrecords       ! records
  integer(SIZE_T), intent(IN) :: type_size       ! type size
  <TYPE>, intent(IN), dimension(*) :: buf        ! data buffer
  integer :: errcode                             ! error code
end subroutine h5tbwrite_field_index_f
```

2.3 Modification

2.3.1 H5TBdelete_record

herr_t **H5TBdelete_record**(hid_t *loc_id*, const char **dset_name*, hsize_t *start*, hsize_t *nrecords*)

`H5TBdelete_record` deletes records from middle of table ("pulling up" all the records after it)

Synopsis Delete records.

Parameters

- **loc_id** – IN: Identifier of the file or group where the table is located.
- **dset_name** – IN: The name of the dataset.
- **start** – IN: The start record to delete from.
- **nrecords** – IN: The number of records to delete.

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

2.3.2 H5TBinsert_record

herr_t **H5TBinsert_record**(hid_t *loc_id*, const char **dset_name*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, void **data*)

`H5TBinsert_record` inserts records into the middle of the table ("pushing down" all the records after it)

Synopsis Insert records.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the table is located.
- **dset_name** – IN: The name of the dataset.
- **start** – IN: The position to insert.
- **nrecords** – IN: The number of records to insert.
- **data** – IN: Buffer with data.

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

2.3.3 H5TBadd_records_from

herr_t **H5TBadd_records_from**(hid_t *loc_id*, const char **dset_name1*, hsize_t *start1*, hsize_t *nrecords*, const char **dset_name2*, hsize_t *start2*)

`H5TBadd_records_from` adds records from a dataset named `dset_name1` to a dataset named `dset_name2`. Both tables are attached to the object specified by the identifier `loc_id`.

Synopsis Add records from first table to second table.

Parameters

- **loc_id** – IN: Identifier of the file or group in which the table is located.
- **dset_name1** – IN: The name of the dataset to read the records.
- **start1** – IN: The position to read the records from the first table
- **nrecords** – IN: The number of records to read from the first table.
- **dset_name2** – IN: The name of the dataset to write the records.
- **start2** – IN: The position to write the records on the second table

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

2.3.4 H5TBcombine_tables

herr_t **H5TBcombine_tables**(hid_t *loc_id1*, const char **dset_name1*, hid_t *loc_id2*, const char **dset_name2*, const char **dset_name3*)

`H5TBcombine_tables` combines records from two datasets named `dset_name1` and `dset_name2`, to a new table named `dset_name3`. These tables can be located on different files, identified by `loc_id1` and `loc_id2` (identifiers obtained with `H5Fcreate`). They can also be located on the same file. In this case one uses the same identifier for both parameters `loc_id1` and `loc_id2`. If two files are used, the third table is written on the first file.

Synopsis Combines records from two tables into a third.

Parameters

- **loc_id1** – IN: Identifier of the file or group in which the first table is located.
- **dset_name1** – IN: The name of the first table to combine.
- **loc_id2** – IN: Identifier of the file or group in which the second table is located.
- **dset_name2** – IN: The name of the second table to combine.
- **dset_name3** – IN: The name of the new table.

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

2.3.5 H5TBinsert_field

herr_t **H5TBinsert_field**(hid_t *loc_id*, const char **dset_name*, const char **field_name*, hid_t *field_type*, hsize_t *position*, const void **fill_data*, const void **data*)

`H5TBinsert_field` inserts a new field named `field_name` into the table `dset_name`. Note: this function requires the table to be re-created and rewritten in its entirety, and this can result in some unused space in the file, and can also take a great deal of time if the table is large.

synopsis Insert a new field into a table.

param loc_id IN: Identifier of the file or group in which the table is located.

param dset_name IN: The name of the table.

param field_name IN: The name of the field to insert.

param field_type IN: The data type of the field.

param position IN: The zero based index position where to insert the field.

param fill_data IN: Fill value data for the field. This parameter can be NULL.

param data IN: Buffer with data.

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

Fortran90 Interface: h5tbinsert_field_f

```
subroutine h5tbinsert_field_f(loc_id, dset_name, field_name, field_type, &
                              field_index, buf, errcode )
  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  character(LEN=*), intent(IN) :: field_name     ! name of the field
  integer(HID_T), intent(IN)   :: field_type     ! field type
  integer, intent(IN) :: field_index             ! field_index
  <TYPE>, intent(IN), dimension(*) :: buf        ! data buffer
  integer :: errcode                             ! error code
end subroutine h5tbinsert_field_f
```

2.3.6 H5TBdelete_field

herr_t **H5TBdelete_field**(hid_t *loc_id*, const char **dset_name*, const char **field_name*)

H5TBdelete_field deletes a field named field_name from the table dset_name. Note: this function requires the table to be re-created and rewritten in its entirety, and this can result in some unused space in the file, and can also take a great deal of time if the table is large.

synopsis Deletes a field from a table.

param loc_id IN: Identifier of the file or group in which the table is located.

param dset_name IN: The name of the table.

param field_name IN: The name of the field to delete.

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

Fortran90 Interface: h5tbdelete_field_f

```
subroutine h5tbdelete_field_f(loc_id, dset_name, field_name, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id        ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name   ! name of the dataset
  character(LEN=*), intent(IN) :: field_name  ! name of the field
  integer :: errcode                          ! error code
end subroutine h5tbdelete_field_f
```

2.4 Retrieval

2.4.1 H5TBread_table

herr_t **H5TBread_table**(hid_t *loc_id*, const char **table_name*, size_t *dst_size*, const size_t **dst_offset*, const size_t **dst_sizes*, void **dst_buf*)

`H5TBread_table` reads a table named `table_name` attached to the object specified by the identifier ``loc_id`.

Synopsis Reads a table.

Parameters

- **loc_id** – IN: Identifier of the file or group to read the table within.
- **table_name** – IN: The name of the dataset to read.
- **dst_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **dst_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET` macro.
- **dst_sizes** – IN: An array containing the sizes of the fields. These sizes can be calculated with the `sizeof()` macro.
- **dst_buf** – OUT: Buffer with data.

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

2.4.2 H5TBread_records

herr_t **H5TBread_records**(hid_t *loc_id*, const char **table_name*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **dst_sizes*, void **data*)

`H5TBread_records` reads some records identified from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

Synopsis Reads records.

Parameters

- **loc_id** – IN: Identifier of the file or group to read the table within.
- **IN** (*table_name*) – The name of the dataset to read.
- **start** – IN: The start record to read from.
- **nrecords** – IN: The number of records to read.
- **type_size** – IN: The size of the structure type, as calculated by `sizeof()`.
- **field_offset** – IN: An array containing the offsets of the fields. These offsets can be calculated with the `HOFFSET` macro.
- **dst_sizes** – IN: An array containing the size in bytes of the fields.
- **data** – OUT: Buffer with data.

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

2.4.3 H5TBread_fields_name

herr_t **H5TBread_fields_name**(hid_t *loc_id*, const char **table_name*, const char * *field_names*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **dst_sizes*, void **data*)

`H5TBread_fields_name` reads the fields identified by `field_names` from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

synopsis Reads one or several fields. The fields are identified by name.

param loc_id IN: Identifier of the file or group to read the table within.

param table_name IN: The name of the dataset to read.

param field_names IN: An array containing the names of the fields to read.

param start IN: The start record to read from.

param nrecords IN: The number of records to read.

param type_size IN: The size in bytes of the structure associated with the table. This value is obtained with `sizeof`.

param field_offset IN: An array containing the offsets of the fields.

param dst_sizes IN: An array containing the size in bytes of the fields.

param data OUT: Buffer with data.

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

Fortran90 Interface: `h5tbread_field_name_f`

```
subroutine h5tbread_field_name_f(loc_id, dset_name, field_name, start, &
                                 nrecords, type_size, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  character(LEN=*), intent(IN) :: field_name     ! name of the field
  integer(HSIZE_T), intent(IN) :: start          ! start record
  integer(HSIZE_T), intent(IN) :: nrecords       ! records
  integer(SIZE_T), intent(IN) :: type_size       ! type size
  <TYPE>, intent(IN), dimension(*) :: buf        ! data buffer
  integer :: errcode                             ! error code
end subroutine h5tbread_field_name_f
```

2.4.4 H5TBread_fields_index

herr_t **H5TBread_fields_index**(hid_t *loc_id*, const char **table_name*, int *nfields*, const int **field_index*, hsize_t *start*, hsize_t *nrecords*, size_t *type_size*, const size_t **field_offset*, const size_t **dst_sizes*, void **data*)

`H5TBread_fields_index` reads the fields identified by `field_index` from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

synopsis Reads one or several fields. The fields are identified by index.

param loc_id IN: Identifier of the file or group to read the table within.

param table_name IN: The name of the dataset to read.

param nfields IN: The number of fields to overwrite. This parameter is also the size of the `field_index` array.

param field_index IN: The indexes of the fields to write.

param start IN: The start record to read from.

param nrecords IN: The number of records to read.

param type_size IN: The size in bytes of the structure associated with the table. This value is obtained with `sizeof`.

param field_offset IN: An array containing the offsets of the fields.

param dst_sizes IN: An array containing the size in bytes of the fields.

param data OUT: Buffer with data.

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

Fortran90 Interface: `h5tbread_field_index_f`

```
subroutine h5tbread_field_index_f(loc_id, dset_name, field_index, start, &
                                  nrecords, type_size, buf, errcode)
  implicit none
  integer(HID_T), intent(IN) :: loc_id           ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name      ! name of the dataset
  integer, intent(IN) :: field_index             ! index
  integer(HSIZE_T), intent(IN) :: start          ! start record
  integer(HSIZE_T), intent(IN) :: nrecords       ! records
  integer(SIZE_T), intent(IN) :: type_size       ! type size
  <TYPE>, intent(IN), dimension(*) :: buf        ! data buffer
  integer :: errcode                             ! error code
end subroutine h5tbread_field_index_f
```

2.5 Query

2.5.1 H5TBget_table_info

herr_t **H5TBget_table_info** (hid_t *loc_id*, const char **table_name*, hsize_t **nfields*, hsize_t **nrecords*)

`H5TBget_dimensions` retrieves the table dimensions from a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

synopsis Gets the table dimensions.

param loc_id IN: Identifier of the file or group to read the table within.

param table_name IN: The name of the dataset to read.

param nfields OUT: The number of fields.

param nrecords OUT: The number of records.

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

Fortran90 Interface: `h5tbget_table_info_f`

```
subroutine h5tbget_table_info_f(loc_id, dset_name, nfields, nrecords, errcode)

  implicit none
  integer(HID_T), intent(IN) :: loc_id        ! file or group identifier
```

```
  character(LEN=*), intent(IN) :: dset_name    ! name of the dataset
  integer(HSIZE_T), intent(INOUT):: nfields    ! nfields
  integer(HSIZE_T), intent(INOUT):: nrecords   ! nrecords
  integer :: errcode                           ! error code
end subroutine h5tbget_table_info_f
```

2.5.2 H5TBget_field_info

herr_t **H5TBget_field_info**(hid_t *loc_id*, const char **table_name*, char **field_names[]*, size_t **field_sizes*, size_t **field_offsets*, size_t **type_size*)

`H5TBget_field_info` gets information about a dataset named `table_name` attached to the object specified by the identifier `loc_id`.

synopsis Gets information about a table.

param loc_id IN: Identifier of the file or group to read the table within.

param table_name IN: The name of the dataset to read.

param field_names OUT: An array containing the names of the fields.

param field_sizes OUT: An array containing the size of the fields.

param field_offsets OUT: An array containing the offsets of the fields.

param type_size OUT: The size of the HDF5 datatype associated with the table. More specifically, the size in bytes of the HDF5 compound datatype used to define a row, or record, in thetable.

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

Fortran90 Interface: `h5tbget_field_info_f`

```
subroutine h5tbget_field_info_f(loc_id, dset_name, nfields, field_names, &
                                field_sizes, field_offsets, type_size, &
                                errcode, maxlen_out)
  implicit none
  integer(HID_T), intent(IN) :: loc_id          ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name     ! name of the dataset
  integer(HSIZE_T), intent(IN):: nfields        ! nfields
  character(LEN=*), dimension(nfields), intent(INOUT) :: field_names
                                                ! field names
  integer(SIZE_T), dimension(nfields), intent(INOUT) :: field_sizes
                                                ! field sizes
  integer(SIZE_T), dimension(nfields), intent(INOUT) :: field_offsets
                                                ! field offsets
  integer(SIZE_T), intent(INOUT):: type_size    ! type size
  integer :: errcode                            ! error code
  integer, optional :: maxlen_out               ! returns maximum character
                                                ! length of field_names
end subroutine h5tbget_field_info_f
```

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

H