RFC: High-Level Functions for Handling Region References and Hyperslab Selections

M. Scot Breitenfeld
Elena Pourmal

This RFC proposes to add new public higher-level C routines to the HDF5 Library for the creation, manipulation, and querying of data associated with a dataset region reference. Proposed high-level routines will reduce the number of steps a user has to transverse when working with region references and will facilitate access to a raw data stored in the NPOESS files.

1 To the reader

New routines address requirements gathered by the HDF5 developers during the years of interaction with developers of the NPOESS file format, with consumers of the future NPOESS data and requests received by The HDF Group HelpDesk. The goal of this document is to engage the NPOESS community of software developers and data consumers in identifying more use cases, gathering and refining requirements and finalizing APIs.

This document is structured as follows: Sections 2-4 give a brief introduction to the HDF5 region references; Section 5 contains description of use cases for the proposed APIs; Section 6 focuses on a general functionality and Section 7 contains Reference Manual entries for the new APIs.

We encourage our readers to send us use cases and identify requirements not covered in this document. We would also like to hear comments on the APIs design and functionality.

2 Introduction to the HDF5 References to Dataset Regions

HDF5 file may contain Gbytes of data organized in a hierarchical structure. An application uses the structure to impose relationships between data objects stored in an HDF5 file, allowing a user to navigate through the data in efficient way. For example, measurements with the same time stamp can be stored in the datasets belonging to the same group in a file. The application just needs to know the path to the group to access all measurements with the same time stamp.

Very often it is not only desired to find related datasets (or other objects), but also to describe a particular subset of data elements stored in the dataset. To achieve this goal, HDF5 provides two special datatypes: references to objects and references to dataset regions.

An object reference points to an HDF5 object, such as a dataset or a group, stored in the same file. The application may use an array of object reference elements to identify a set of related HDF5 objects.

A region reference points to a dataset and a region within that dataset. When stored in a dataset, an array of region references can provide a unified view of the data stored in the different datasets in a file.

In general, region references are useful for directly accessing a portion of a dataset. Notably, region references play an important role in large datasets by providing a convenient and efficient way to point to data of interest.

Figure 1 illustrates the concept of region references. A user can create a file, *FileA.h5*, having a group (*Group_1D*) containing data values stored in a one dimensional array (*DS1*), a group (*Group_2D*) containing data in a two dimensional array (*DS2*), and a group (*Group_3D*) containing data in a three dimensional array (*DS3*). In order to quickly and efficiently access data within a dataset, an array of region references is created (*R1*). Each element in the region reference array points to a different selection of elements in the datasets.

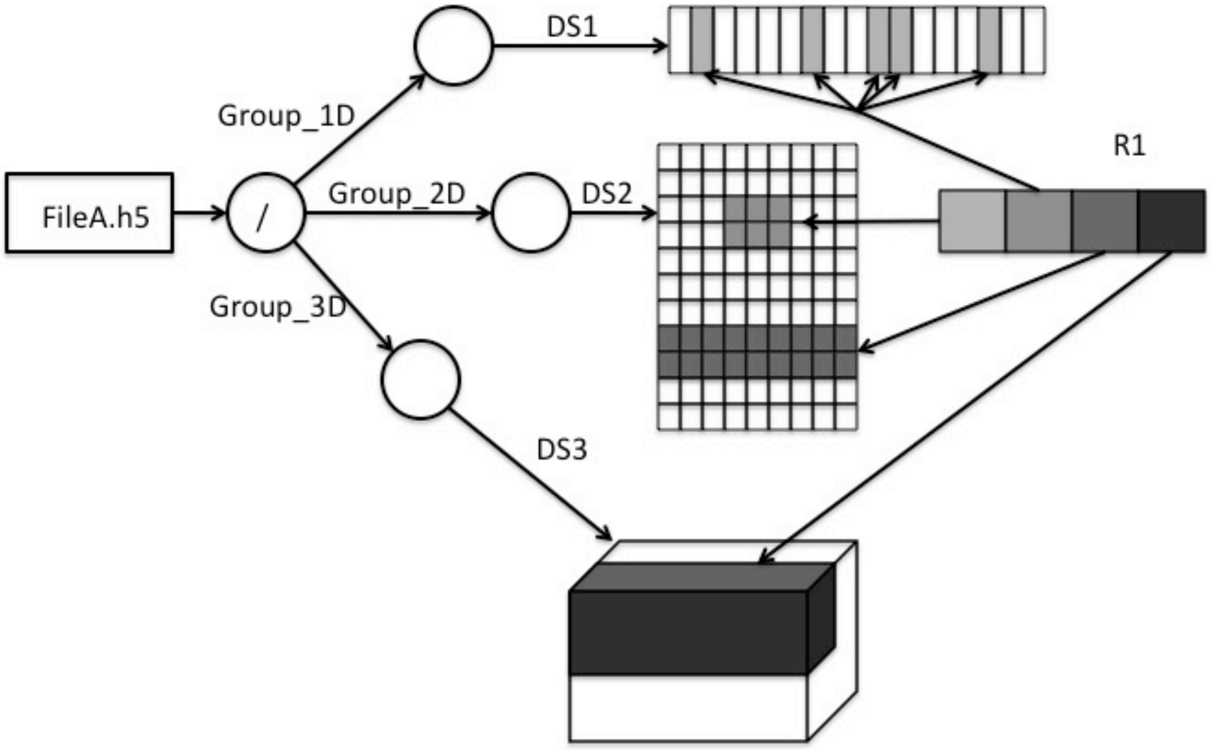

Figure 1

Below is an output of the h5dump utility (version 1.8) run on the FileA.h5 file depicted above. In the output the three groups are listed along with each group's dataset and the dataset R1 with region references. Each element of R1 is displayed as a path to a referenced dataset and a description of the selected elements. The point selection in the one-dimensional dataset DS1 is displayed as a list of the selected elements' coordinates; selections in the two and three-dimensional datasets, DS2 and DS3, are displayed as lists of hyperslabs' upper left and lower right corner coordinates. Coordinates of the elements are 0-based.

```
HDF5 "FileA.h5" {
GROUP "/" {
   GROUP "Group_1D" {
      DATASET "DS1" {
         DATATYPE  H5T_STD_I32LE
         DATASPACE  SIMPLE { ( 17 ) / ( 17 ) }
         DATA { …
                  }
      }
   }
   GROUP "Group_2D" {
      DATASET "DS2" {
         DATATYPE  H5T_STD_I32LE
         DATASPACE  SIMPLE { ( 10, 9 ) / ( 10, 9 ) }
         DATA { …
         }
      }
   }
   GROUP "Group_3D" {
      DATASET "DS3" {
         DATATYPE  H5T_STD_I32LE
         DATASPACE  SIMPLE { ( 6, 6, 6 ) / ( 6, 6, 6 ) }
         DATA { …
         }
      }
   }
   DATASET "R1" {
      DATATYPE  H5T_REFERENCE
      DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
      DATA {
(0):          DATASET /Group_1D/DS1 {(1), (6), (9), (10), (14)},
(1):          DATASET /Group_2D/DS2 {(3,3)-(5,4)},
(2):          DATASET /Group_2D/DS2 {(0,7)-(8,8)},
(4):          DATASET /Group_3D/DS3 {(0,0,2)-(5,2,5)}
      }
   }
}
}
```

Figure 1A

An element of a region reference datatype may also be an attribute of a dataset or a group. For instance, if the data were a series of velocity measurements at various water depths then the attribute shown in Figure 2 at each depth would contain a region reference pointing to the proper data for that depth.

|

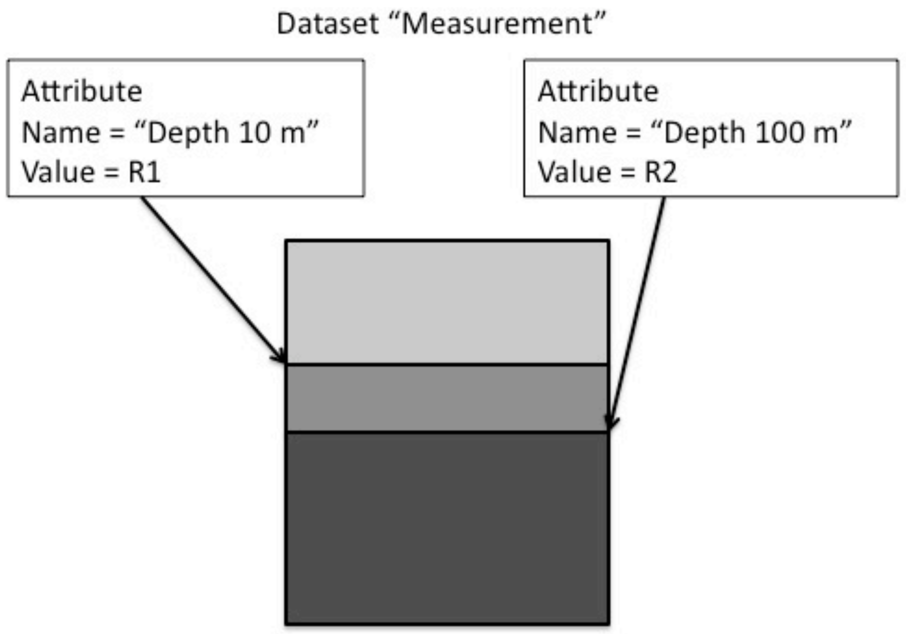

Figure 2

3 NPOESS Example

A real world example relying heavily on the HDF5 object and region references is the National Polar-Orbiting Operational Environmental Satellite System (NPOESS) project, which monitors environmental conditions of the earth. Many NPOESS products (RDRs, SDRs, EDRs[1]) will be distributed in HDF5 files with a special file structure that consists of a user block with some human readable information, and two groups containing datasets with raw data and datasets with objects and region reference data that constitute a particular data product. For example, each NPOESS RDR file will contain an "aggregation" dataset. Elements of an "aggregation" dataset point to the datasets with raw data that *belong to a product*. A "granule" dataset in an NPOESS file represents sensor scans taken during a time interval. Each element of a granule dataset has a region reference datatype and points to a dataset with raw data and a contiguous subset within the data.

[1] Raw Data Record (RDR), Sensor Data Record (SDR), Environmental Data Record (EDR)

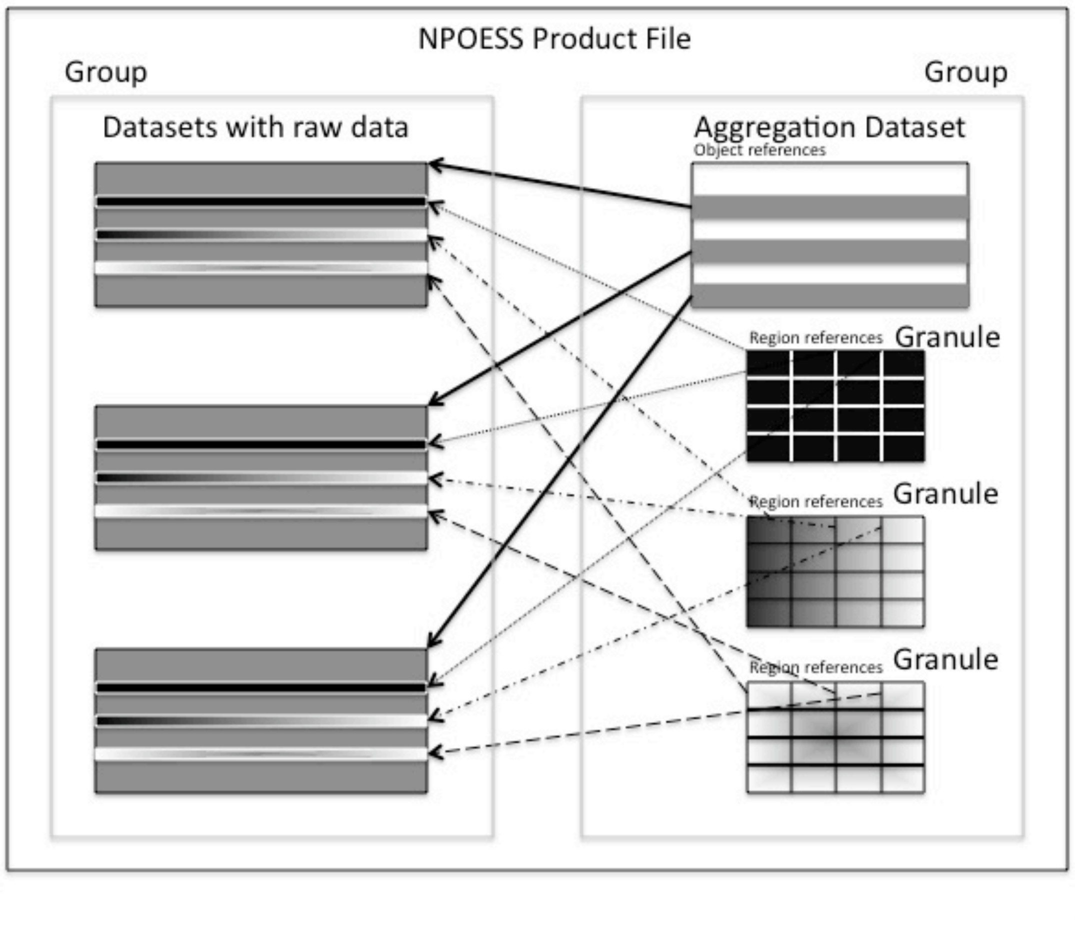

Figure 3

4 Operations on the HDF5 Region References and their limitations

The HDF5 library provides functions in the H5R interface to handle HDF5 object and dataset region references. Those functions enable:

- Creation of an object and dataset region reference
- Obtaining an identifier of the referenced object
- Obtaining an identifier of the referenced dataset
- Obtaining an identifier of the dataspace selection corresponding to a region within a referenced dataset

All H5R APIs use low-level information such as object identifiers, internal representations of objects and dataset region references to access and retrieve the data the references point to. Information such as a path to the referenced object, characteristics of a dataset region (such as its size, its type and the number of elements in it), the position of the region within the data array, and the data itself have to be retrieved by an application using functions from different HDF5 interfaces: H5R, H5D, H5T, and H5S. For example, h5dump utility uses H5Rget_name, H5Sselect_type, H5Sselect_npoints, H5Sselect_hyper_nblocks, and H5Sselect_hyper_blocklist, among other APIs, to retrieve and print paths to datasets and descriptions of selections (see h5dump output above). h5dump doesn't display raw data corresponding to the dataset regions. Currently, this functionality is not provided by any of the HDF5 tools and requires considerable programming effort.

The next two sections of this document describe different use cases for HDF5 dataset region references and propose several high-level APIs that will facilitate development of applications working with data region references. The primary goal of these new routines is to avoid an otherwise substantial programming effort for data discovery and retrieval.

5 Use cases for High-Level Dataset Region References and Hyperslab APIs

This section describes several use cases that illustrate different ways of how the users would access data associated with the region references.

Reading granule's data from NPOESS file

1. A weather simulation application needs raw data from an NPOESS RDR file to initialize input parameters before running a computational model. The application opens an NPOESS RDR file and a granule dataset specified by the User, and reads raw data from multiple datasets in the RDR file pointed to by the region references in the granule dataset. The data, as in Figure 4, is returned to a buffer allocated by the application.
2. A User receives an NPOESS EDR file that contains several granules. He is interested only in one granule and the corresponding raw data that he would like to store in another HDF5 file. The User uses an NPExport tool[2] to copy the granule's raw data to a dataset in a specified HDF5 file. He also uses other features in NPExport to copy any appropriate attributes and quality flags.

HDF5 Tools

1. A User runs the h5dump utility with a command line flag to display header information for all datasets with the region references in an HDF5 file.

2. After examining the output, the User chooses the dataset(s) of interest and runs h5dump with a command line flag to display region reference data elements as an absolute path to a referenced dataset, as well as a list of corner coordinates for the corresponding region (hyperslab selection).

[2] The actual tool doesn't exist; it is used for "use case" purposes only

3. After examining the output and choosing a dataset and a region of interest, the User provides h5dump with the absolute path to a dataset and a list of corner coordinates to display the referenced data. H5dump displays header information for the referenced dataset and raw data from the *referenced region.*

4. A User runs the h5dump utility on an HDF5 file and a dataset with region references using a specified flag(s) to display header information for the referenced dataset and *raw data from the referenced region for each element of the dataset with region references.* (Note: compare with steps 2-3 above; raw data will be displayed automatically in this case.)

5. For big datasets with region references it is not practical to display the raw data for *all* region references. In this case, a User runs the h5dump using a specified flag(s) utility only on a *subset* of a dataset with region references to display header information for the referenced datasets and raw data from the referenced regions.

6. Cases 1-5 applied to h5ls

7. A User runs the h5copy utility to copy a dataset with region references and all datasets the region references point to, to the locations in the new file specified by the User. H5copy finds all referenced datasets in the source file, copies them to the destination file, creates new region references for the newly copied datasets and corresponding regions, and writes a dataset with the region references in a destination file.

8. A User wants to copy part of one dataset to another. He specifies a path to a source dataset and corner coordinates of a hyperslab and a path to a destination dataset and corner coordinates of a destination hyperslab to the h5copy utility. The tool copies the raw data from one dataset to another. (Note: This scenario doesn't use region references, but rather explores how the hyperslab description obtained from a region reference can be useful in retrieving and saving the raw data of interest.)

9. A User starts HDFView and using a special icon sets properties to show datasets with region references. He opens the HDF5 file and inspects the datasets. The User finds the dataset of interest and clicks the right button to display a menu with data display options. He chooses to display a region reference data in a spreadsheet with each element shown as a path to a dataset and corner coordinates of a hyperslab. The User opens the dataset with region references, examines the data and clicks on one of the elements. HDFView spawns a spreadsheet window with the referenced data displayed.

10. A User starts HDFView and sets properties to show datasets with region references using a special icon. He opens the HDF5 file and inspects the datasets. The User finds the dataset of interest and clicks the right button to display a menu with data display options. He chooses to display the region reference data graphically. The User opens the dataset with region

references. HDFView spawns another window with the file tree showing all referenced datasets (i.e., datasets referenced by the elements of the dataset with region references). The User clicks on one of the datasets. HDFView displays data from the referenced region in a spreadsheet.

11. A User opens a dataset as an image in HDFView. He selects a sub-region by clicking and dragging the mouse. Then he uses one of the menus to create a region reference and store it in a buffer. The User opens another dataset, selects a region, creates the second reference, etc. When all desired region references are created the user saves them in a dataset in the file. Now he can use the dataset with region references to find the datasets and regions of interest.

6 Approach

The purpose of the higher-level region reference functions and functions to read/write hyperslabs of data described below is to combine various HDF5 APIs into a single API, which is then used for querying, creating, accessing and manipulating data associated with a region reference.

For now we assume that a region reference points to a rectangular sub-array of data that can be described by the coordinates of its upper left and lower right corners. Generalization for more complex selections such as subsets created by union and difference set operations or element selections can be done later with a slight change to the API signatures (see the next section below) and without changing the functionality this proposal focuses on.

The new APIs are designed to handle the following set of operations associated with the dataset region references and hyperslab selections.

Query information

The query API facilitates the discovery of the referenced dataset (i.e., the dataset that a region reference points to), its datatype, dimensionality and the description of the selected region using corners coordinates. This API will provide information similar to the information displayed by the h5dump utility (see Figure 1A). Currently several HDF5 routines from H5D, H5I and H5S interfaces have to be called to extract a path to a dataset, a dataset's datatype and hyperslab selection description using information returned by H5R APIs.

Proposed new function `H5LRget_region_info` will use an element of region reference datatype to return

a. The full path to the referenced dataset.

b. The description of the hyperslab selection the reference points to (i.e. list of corners).

c. The file datatype of the referenced data.

d. The rank of the dataset.

e. The type of selection (point or hyperslab selection).

Access region reference raw data

Three new reading APIs will allow retrieving referenced data to a buffer specified by a user or to write it directly to a dataset bypassing several calls to the APIs from H5R, H5D, H5T and H5S interfaces. Access APIs will use either a region reference or a path and hyperslab corner coordinates returned by a query API (or specified by a user to retrieve the data). Proposed APIs will allow an application to perform datatype conversion on the fly as the current HDF5 I/O APIs do.

The first proposed new function `H5LRread_region` (see Figure 4) uses an element of a region reference to return

a. The number of elements in the selected region

b. Data into a buffer allocated by the application

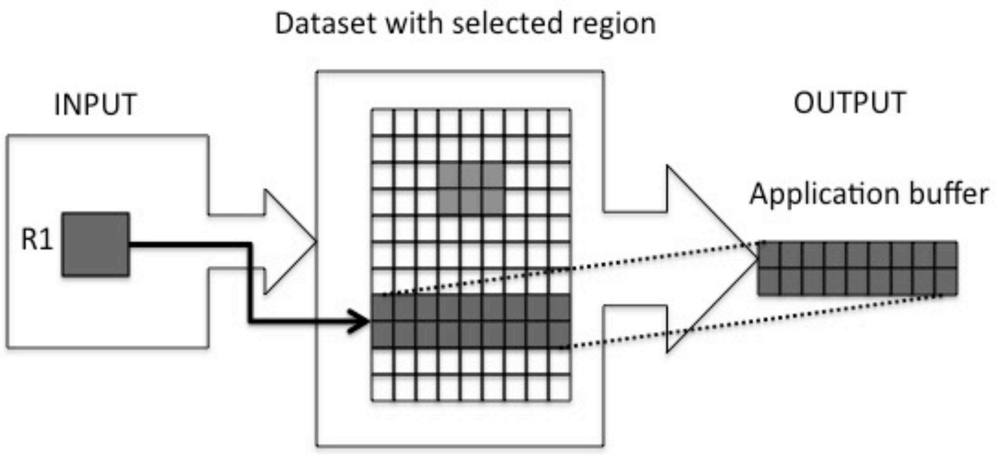

Figure 4

The second proposed new function `H5LTread_region` (see Figure 5) has similar functionality, but does not use a region reference as input. It uses a path to a dataset and hyperslab corner coordinates (for example, obtained by the `H5LRget_region_info` function) to return

a. The number of elements in the selected region

b. Data into a buffer allocated by the application

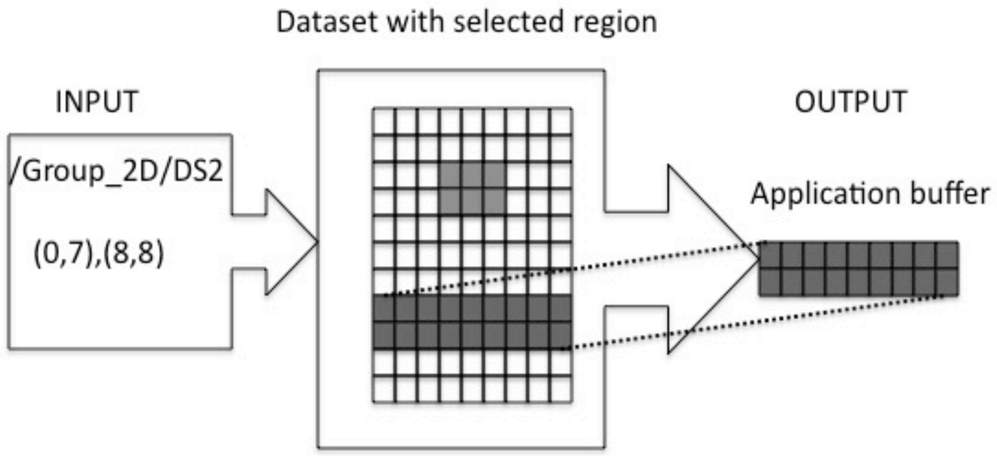

Figure 5

As with all general high-level functions it has the prefix `H5LT` instead of `H5LR` to emphasize the difference.

The third proposed new function `H5LRmake_dataset` creates a new dataset from the regions that an array of region references points to. For example, a dataset can be created that contains all the data belonging to one granule, using an array of region references (see Figure 6). It will be assumed that all regions have the same rank and dimensions sizes except the slowest changing one.

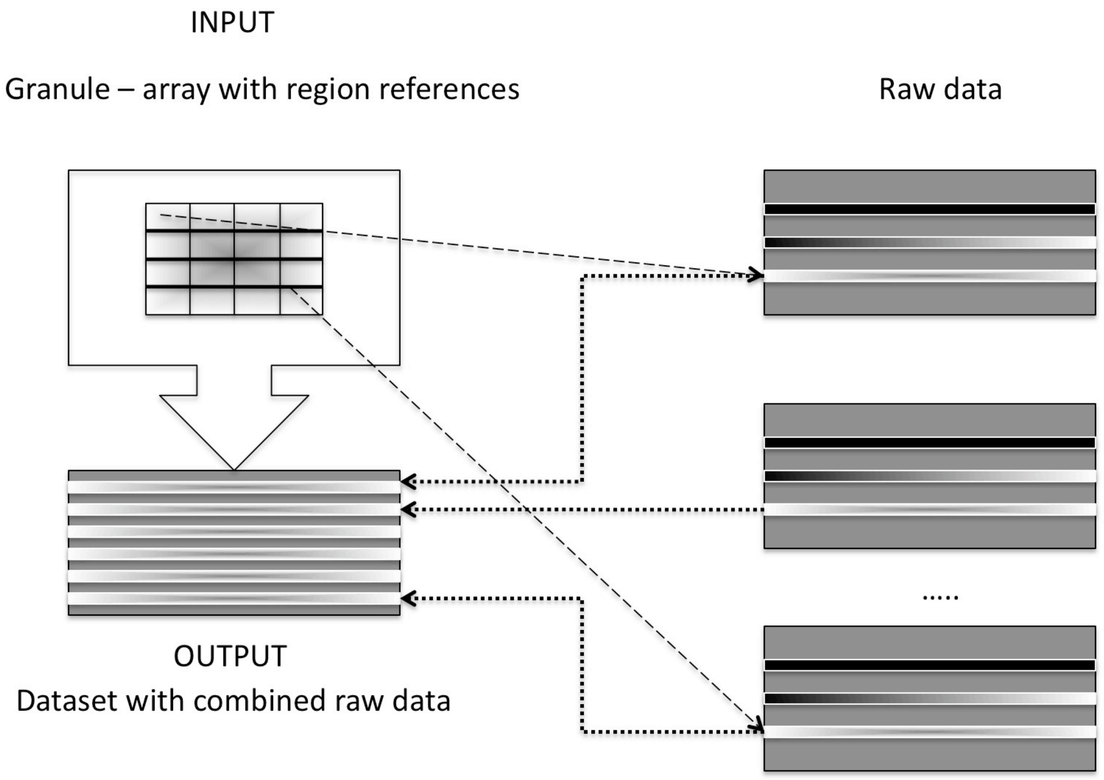
INPUT
Granule – array with region references
Raw data
.....
OUTPUT
Dataset with combined raw data

Figure 6

Create/write data associated with region references

This section describes four new APIs for creating and writing region references and data.

The current process of creating datasets with region references is very tedious: a region reference element is created one at a time using HDF5 library information such as dataset and dataspace identifiers, then stored in an application buffer which is then written to a dataset. The new APIs will allow one to easily create arrays of references using high-level descriptions of the selected regions.

The proposed API `H5LRcreate_region_references` will use a list of paths to datasets and a list of corresponding hyperslab descriptions (corner coordinates) to *create an array of region references in a buffer provided by an application* (see Figure 7). This new API is useful for applications like HDFView: it will allow users to save identified regions of interest by creating an array of region references and then saving it into a dataset or an attribute in an HDF5 file. It can also be used for updating datasets with region references when objects are deleted in a file or copied from one file to another.

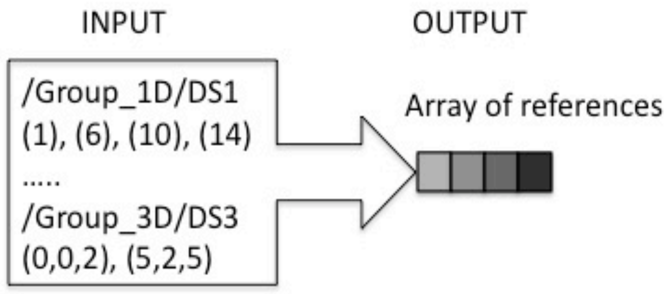

Figure 7

The next two APIs will help to copy *raw data* pointed by a region reference. One can use them, for example, for collecting data stored in *different* datasets into one.

Currently those operations involve many steps, from discovering datasets and regions pointed to by region references to reading data into an application buffer and writing it to a specified region of another dataset. The functions proposed below will facilitate this process tremendously.

The first proposed function `H5LRcopy_region` writes data pointed to by a region reference to a dataset region specified by an application (see Figure 8). The function will use a path to a dataset and hyperslab corner coordinates to describe the new location of the data.

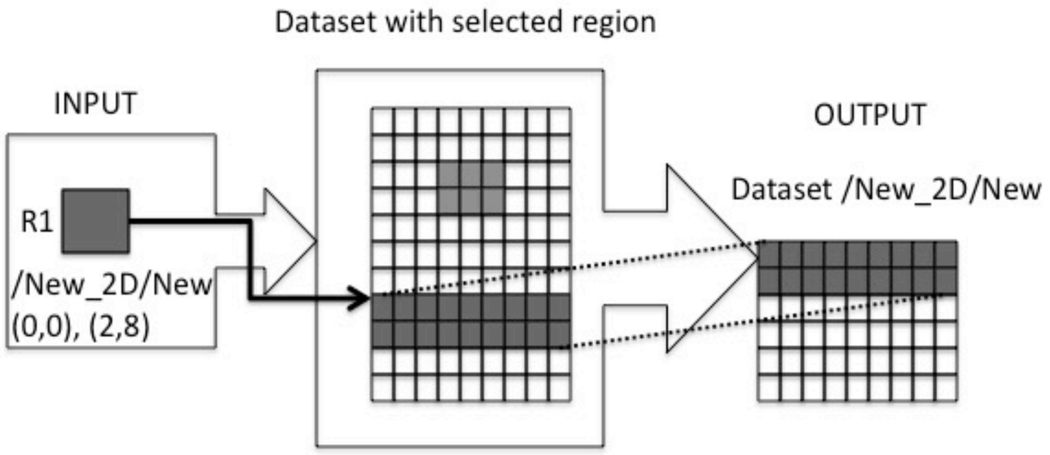

Figure 8

The second proposed function `H5LTcopy_region` will have the same functionality as the previous one, but use a path to a dataset and corner coordinates of the hyperslab to describe the referenced data (source data) rather than region reference as shown in Figure 9. Therefore it has prefix `H5LT` instead of `H5LR` to emphasize the difference.

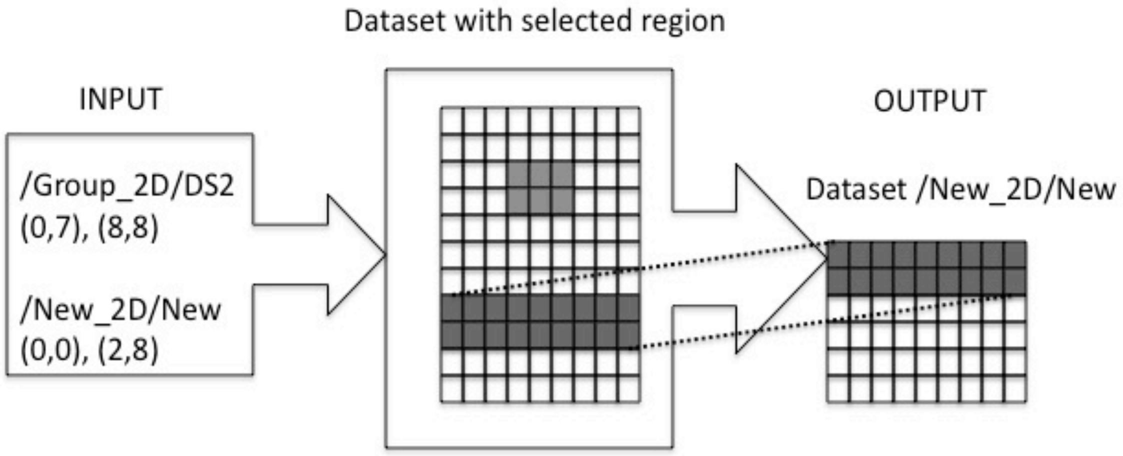

Figure 9

The last proposed API routine `H5LRcopy_reference`, would copy data pointed to by the dataset region reference to a new location and create a dataset region reference to it (see Figure 10).

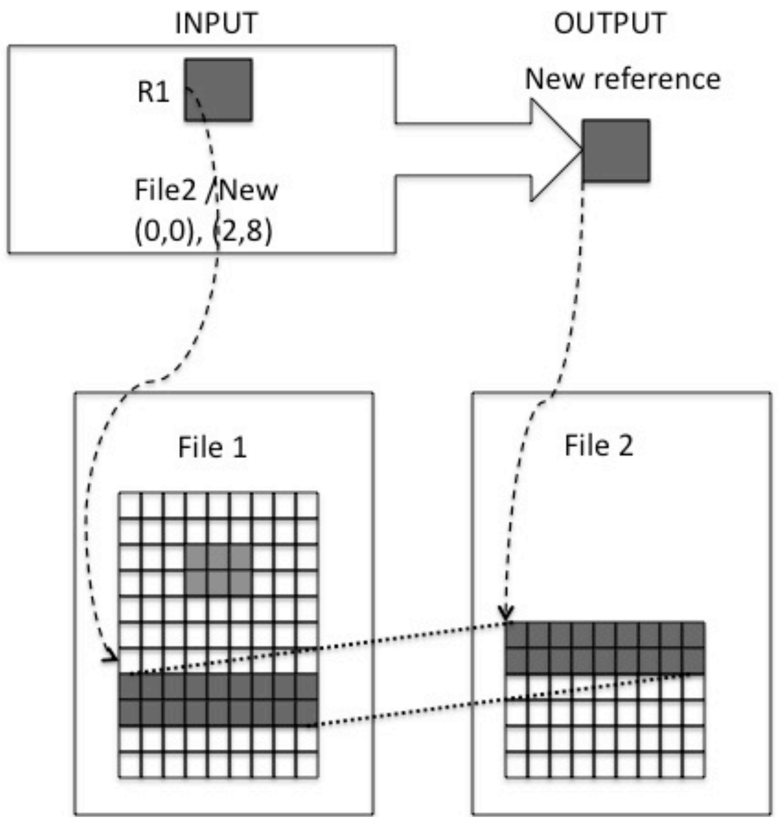

Figure 10

7 Sample Reference Manual Entries

The functions below assume that a region reference is a simple hyperslab described by its upper left and low right corners. Those APIs can be extended to handle hyperslabs that are results of the set operations on the simple hyperslabs, or to point selections.

Region reference APIs

Name: `H5LRget_region_info`

Signature:

herr_t `H5LRget_region_info(` *hid_t* `obj_id`, *const hdset_reg_ref_t* `*ref`, *size_t* `*len`, *char* `*path`, *int* `*rank`, *hid_t* `*dtype` , *H5S_sel_type* `*sel_type`, *size_t* `*numelem`, *size_t* `*buf`)

Purpose:

Retrieves information about the data a region reference points to.

Description:

`H5LRget_region_info` queries information about the data pointed by a region reference `ref`. It returns one of the absolute paths to a dataset, length of the path, dataset's rank and datatype, description of the referenced region and type of the referenced region. Any output argument can be `NULL` if that argument does not need to be returned.

The parameter `obj_id` is an identifier for any object in the HDF5 file containing the referenced object. For example, it can be an identifier of a dataset the region reference belongs to or an identifier of an HDF5 file the dataset with region references is stored in.

The parameter `ref` is a region reference to query.

The parameter `path` is a pointer to application allocated buffer of size `len+1` to return an absolute path to a dataset the region reference points to.

The parameter `len` is a length of absolute path string. If `path` parameter is `NULL`, actual length of the path is returned to application and can be used to allocate buffer `path` of an appropriate length `len+1`.

The parameter `sel_type` describes the type of the selected region. Possible values can be `H5S_SEL_POINTS` for point selection and `H5S_SEL_HYPERSLABS` for hyperslab selection.

The parameter `numelem` describes how many elements will be placed in the buffer `buf`. The number should be interpreted using the value of `flag`.

If value of `flag` is `H5S_SEL_HYPERSLABS`, the parameter `buf` contains `numelem` blocks of the coordinates for each simple hyperslab of the referenced region. Each block has length `2*rank` and is organized as follows: <"start" coordinate>, immediately followed by <"opposite" corner coordinate>[3]. The total size of the buffer to hold the description of the region will be `2*rank*numelem`. If region reference points to a contiguous sub-array, then the value of `numelem` is 1 and the block contains coordinates of upper left and lower right corners of the sub-array (or simple hyperslab).

If value of `flag` is `H5S_SEL_POINTS`, the parameter `buf` contains `numelem` blocks of the coordinates for each selected point of the referenced region. Each block has length `rank` and contains coordinates of the element. The total size of the buffer to hold the description of the region will be `rank*numelem`.

[3] This parameter follows the convention for hyperslab descriptions used by the H5Sselect_hyper_nblocks and H5Sselect_hyper_blocklist functions. Maybe it will be more convenient to use description as $(u_1, u_2,\dots,u_n), (l_1,l_2,\dots,l_n)$ rather then $(u_1, l_1), (u_2, l_2),\dots, (u_n, l_n)$ as it is done now.

Parameters:

hid_t `obj_id`	IN: Identifier of any object in an HDF5 file the region reference belongs to.
const hdset_reg_ref_t `*ref`	IN: Region reference to query.
size_t `*len`	IN/OUT: size of the buffer path
char `*path`	OUT: full path that a region reference points to.
int `*rank`	OUT: the number of dimensions of the dataset pointed by region reference.
hid_t `*dtype`	OUT: datatype of the dataset pointed by the region reference
H5S_sel_type `*sel_type`	OUT: type of the selection (point or hyperslab)
size_t `*numelem`	IN/OUT: number of coordinate blocks or selected elements
size_t `*buf`	OUT: buffer containing description of the region pointed by region reference

Returns:

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

Notes:

This API can be replaced by several APIs if the list of parameters is too long. For example, we can create an API that will just return a path and another API will return the description of the referenced region.

The first parameter can be just a name of the file or any other object in that file. We use object identifier here as original H5R APIs.

Name: `H5LRread_region`

Signature:

herr_t `H5LRread_region` (*hid_t* `obj_id`, *const hdset_reg_ref_t* `*ref`, *hid_t* `mem_type`, *size_t* `*numelem`, *void* `*buf`)

Purpose:

Read raw data pointed by a region reference to an application buffer.

Description:

The function `H5LRread_region` reads data pointed to by a region reference into a buffer `buf` allocated by an application. The buffer should be big enough to hold `numelem` elements of the type that corresponds to the `mem_type` datatype. For example, if data is read from the referenced region into an integer buffer, `mem_type` should be `H5T_NATIVE_INT` and the size of the buffer should be at least `sizeof(int)*numelem` bytes in size.

The parameter `obj_id` is an identifier for any object in the HDF5 file containing the referenced object. For example, it can be an identifier of a dataset the region reference belongs to or an identifier of an HDF5 file the dataset with region references is stored in.

The parameter `ref` is a region reference to query.

The parameter `mem_type` is an identifier of the HDF5 datatype that describes data in an application buffer.

The parameter `numelem` is number of elements will be read into buffer `buf`. When the size of the buffer is unknown, the API can be called with the `buf` parameter set to NULL in order to retrieve the number of elements in a referenced region.

Parameters:

hid_t `obj_id`	IN: Identifier of any object in a file an HDF5 reference belongs to.
Const hdset_reg_ref_t `*ref`	IN: Region reference to query.
hid_t `mem_type`	IN: memory datatype; describes the buffer the referenced data will be read into
hsize_t `*numelem`	IN/OUT: number of elements in the referenced region
void `*buf`	OUT: buffer containing data from the referenced region

Returns:

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

Name: `H5LRcreate_region_references`

Signature:

herr_t `H5LRcreate_region_references`(*hid_t* `obj_id`, *size_t* `numelem`, *const char* `**path`, *const hsize_t* `*block_coord`, *hdset_reg_ref_t* `*buf`)

Purpose:

Create an array of region references using an array of paths to datasets in a file and an array of the corresponding hyperslab descriptions.

Description:

H5LRcreate_region_references is useful for creating a list of region references given a list of paths to the datasets and corner coordinates of the corresponding hyperslabs. It is assumed that all hyperslabs have the same dimensionality rank. The path parameter is an array of pointers to strings. The parameter numelem is a size of the path and ref arrays.

Buffer block_coord contains descriptions of the hyperslabs following the format <"start" coordinate>, immediately followed by <"opposite" corner coordinate> "rank" times for each numelem hyperslabs. Please note that rank may vary from one dataset to another. The function will retrieve the rank for each dataset and will use the values to interpret the values in the block_coord buffer.

Parameters:

hid_t obj_id	IN: Identifier of any object in a file
size_t numelem	IN: Number of elements in path and buf arrays
const char **path	IN: Array of pointers to strings
const hsize_t *block_coord	IN: Array of hyperslabs coordinates
hdset_reg_ref_t *buf	OUT: Array of region references

Returns:

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

Name: H5LRmake_dataset

Signature:

herr_t H5LRmake_dataset (*hid_t* loc_id, *const char* *path,
hid_t type_id, *const size_t* numelem, *const hid_t* *obj_id, *const hdset_reg_ref_t* *buf)

Purpose:

Creates a dataset and writes data associated with a list of region references to it.

Description:

Given an array of size numelem of region references buf, the function will create a dataset with path path, at location specified by loc_id and of a datatype specified by type_id, and will write data associated with each region reference in the order corresponding to the order of the region references in the buffer. It is assumed that all referenced hyperslabs have the same dimensionality, and only the size of the slowest changing dimension may differ. Each reference in the buf array belongs to the file identified by the corresponding object identifiers in the array obj_id.

Parameters:

hid_t loc_id	IN: Location identifier of the dataset to be created
const char *path	IN: Path to the dataset
hid_t type_id	IN: Datatype of the dataset
const size_t numelem	IN: Size of the obj_id and buf arrays
const hid_t *obj_id	IN: Array of object identifiers; each identifier describes to which HDF5 file the corresponding region reference belongs to
const hdset_reg_ref_t *buf	IN: Array of region references

Returns:

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

Name: H5LRcopy_region

Signature:

herr_t H5LRcopy_region(*hid_t* obj_id, *hdset_reg_ref_t* *ref, *const char* *file_dest, *const char* *path_dest, *const hsize_t* *block_coord_dest)

Purpose:

Copies data from a region specified by a reference to a region in a destination dataset.

Description:

Given a dataset region reference ref in a source file specified by an identifier of any object in that file obj_id, the function will write data to the existing dataset path_dest in file file_dest to the simple hyperslab specified by block_coord_dest. The array block_coord_dest has length 2*rank and is organized as follows: <"start" coordinate>, immediately followed by <"opposite" corner coordinate>.

Parameters:

hid_t obj_id	IN: Identifier of any object in a file dataset region reference belongs to
hdset_reg_ref_t *ref	IN: Dataset region reference
const char *file_dest	IN: Name of the destination file
const char *path_dest	IN: Full path to the destination dataset

const hsize_t *block_coord_dest	IN: Hyperslab coordinates in the destination dataset

Returns:

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

Name: `H5LRcopy_reference`

Signature:

herr_t `H5LRcopy_reference`(*hid_t* `obj_id`, *hdset_reg_ref_t* `*ref`, *const char* `*file`, *const char* `*path`, *const hsize_t* `*block_coord`, *dset_reg_ref_t* `*ref_new`)

Purpose:

Copy data from the specified dataset to a new location and create a reference to it.

Description:

Given a data set pointed to by a region reference, the function `H5LRcopy_reference` will copy the hyperslab data referenced by a datasets region reference into existing dataset specified by its path `path` in the file with the name `file`, and to location specified by the hyperslab coordinates `block_coord`. It will create the region reference `ref_new` to point to the new location. The number of elements in the old and newly specified regions has to be the same.

Parameters:

hid_t `obj_id`	IN: Identifier of any object in a file an HDF5 reference belongs to.
hdset_reg_ref_t `ref`	IN: Reference to the datasets region
const char `*file`	IN: Name of the destination file
const char `*path`	IN: Full path to the destination dataset
const hsize_t `*block_coord`	IN: Hyperslab coordinates in the destination dataset
hdset_reg_ref_t `*ref_new`	OUT: Region reference to the new location of data

Hyperslab APIs

Name: `H5LTread_region`

Signature:

herr_t `H5LTread_region` (*const char* `*file`, *const char* `*path`, *const hsize_t* `*block_coord`, *hid_t* `mem_type`, *void* `*buf`)

Purpose:

Read selected data to an application buffer.

Description:

`H5LRread_region` reads data from a region described by the hyperslab coordinates in `block_coord`, located in the dataset specified by its absolute path `path` in a file specified by its name `file`. Data is read into a buffer `buf` of the datatype that corresponds to the HDF5 datatype specified by `mem_type`.

Buffer `block_coord` has size `2*rank` and contains description of the hyperslab following the format <"start" coordinate>, immediately followed by <"opposite" corner coordinate>.

Buffer `buf` should be big enough to hold selected elements of the type that corresponds to the `mem_type` datatype.

Parameters:

const char `*file`	IN: Name of the file
const char `*path`	IN: Full path to a dataset
const hsize_t `*block_coord`	IN: hyperslab coordinates
hid_t `mem_type`	IN: memory datatype; describes the buffer the referenced data will be read into.
void `*buf`	OUT: buffer containing data from the referenced region

Returns:

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

Name: `H5LTcopy_region`

Signature:

herr_t `H5LTcopy_region`(*const char* `*file_src`, *const char* `*path_src`, *const hsize_t* `*block_coord_src`, *const char* `*file_dest`, *const char* `*path_dest`, *const hsize_t* `*block_coord_dest`)

Purpose:

Copy data from a specified region in a source dataset to a specified region in a destination dataset.

Description:

Given a path to a dataset `path_src` in a file with the name `file_src`, and description of a simple hyperslab of the source `block_coord_src`, the function will write data to the dataset `path_dest` in file `file_dest` to the simple hyperslab specified by `block_coord_dest`. Each `block_coord_src` and `block_coord_dest` array has length `2*rank` and is organized as follows: <"start" coordinate>, immediately followed by <"opposite" corner coordinate>.

Parameters:

const char `*file_src`	IN: Name of the source file
const char `*path_src`	IN: Full path to the source dataset
const hsize_t `*block_coord_src`	IN: Hyperslab coordinates in the source dataset
const char `*file_dest`	IN: Name of the destination file
const char `*path_dest`	IN: Full path to the destination dataset
const hsize_t `*block_coord_dest`	IN: Hyperslab coordinates in the destination dataset

Returns:

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

8 Code Example

This code demonstrates how raw data associated with NPOESS granule can be retrieved and stored in an application specified dataset. The program opens an NPOESS file and a granule. H5Dread is used to read data from the granule dataset. `H5LRmake_dataset` call is used to store associated raw data in a new dataset.

```
#include <hdf5_hl.h>

main()
{
 …
   /*
    * Open NPOESS product file and a granule dataset.
    */
    file = H5Fopen (“NPOESS RDR”, H5F_ACC_RDONLY, H5P_DEFAULT);
    dset = H5Dopen (file, “Granule 1”);

    /*
     * Get dataspace and allocate memory for read buffer.
     */
    space = H5Dget_space (dset);
    ndims = H5Sget_simple_extent_dims (space, dims, NULL);
    rdata = (hdset_reg_ref_t *) malloc (dims[0] * sizeof (hdset_reg_ref_t));
    status = H5Sclose (space);

    /*
     * Read the region references
     */
    status = H5Dread (dset, H5T_STD_REF_DSETREG, H5S_ALL, H5S_ALL, H5P_DEFAULT,
                rdata);
     /*
      * Write raw data associated with the granule. This call will create
      * new dataset with a user specified datatype and write raw data to it.
      */
    status = H5LRmake_dataset (file_new, “My data”, H5T_IEEE_F64BE, rdata);
  …


 }
```
.

References

1. “Profile of National Polar-Orbiting Operational Satellite System (NPOESS) HDF5 Files”, *Kim Tomashosky, Ken Stone, Pat Purcell, Ron Andrews, HDF and HDF-EOS Workshop X, 2006, Landover, Maryland,* *http://www.hdfeos.net/workshops/ws10/presentations/day3/Profile_of_NPOESS_HDF5_Files.ppt*
2. “NPP/ NPOESS Product Data Format”, *Richard E. Ullman, HDF and HDF-EOS Workshop XI, 2007, Landover, Maryland,* *http://www.hdfeos.net/workshops/ws11/presentations/day2/NPOESS-Format-Talk.ppt*
3. “HDF Group Support for NPP/NPOESS”, *Mike Folk, HDF and HDF-EOS Workshop XII, 2008, Aurora, Colorado, http://www.hdfeos.net/workshops/ws12/presentations/day3/mxf.ppt*

Acknowledgements

This work is supported by the NPOESS project.

Revision History

March 14, 2009:	Version 1 circulated for comment within The HDF Group.
March 23, 2009:	Version 2 circulated for comment within The HDF Group.
March 26, 2009:	Version 3 circulated for comment within NPOESS developers.

Comments should be sent to epourmal@hdfgroup.org or help@hdfgroup.org