HDF EOS Follow up

The HDF Group
June 8, 2005

This note discusses several issues in the HDF ESDIS meeting held in April.

1. HDF-EOS2 release: update SZIP, HDF4 libraries

HDF-EOS 2.13 was released with HDF4.2r0 and SZIP1.2. We strongly recommend that all users move to SZIP 2.0 and HDF4.2r1.

SZIP compression was nearly completely broken in HDF4.2r0.[1] Users should upgrade to HDF4.2r1 as soon as possible. Any data compressed with SZIP using HDF4.2r0 may or may not be accessible.

In HDF4.2r1, the API for enabling SZIP compression for an HDF SDS was simplified.

Also, Version 2.0 of SZIP includes a new feature that makes it easy to configure the SZIP library with the encoder disabled. This makes it easy for software providers to distribute the SZIP decoder, which is free for all users. (The decoder requires a license for commercial use.)

A new API for SZIP and in HDF4 enables programs to determine whether SZIP is configured with decoder-only or encoder and decoder.

Please see the following information about SZIP compression in HDF4.2r1:
http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/SZIP_HDF4_2r1.pdf

2. Future HDF-EOS release: HDF4 libraries for version 4.2r2

Important notice: the HDF group is planning to change the names and contents of the HDF4 libraries from libdf and libmfhdf to libhdf4 and libhdf4_fortran in HDF4.2r2

The HDF4 configuration was reworked for HDF4.2r0. Starting with that release, HDF4 configuration uses GNU autotools. GNU autotools support in HDF4 reduces maintenance costs, facilitates porting to the new platforms, and makes it possible to support HDF4 C shared libraries. Support for the HDF4 shared libraries has been requested by many users and is also needed to provide seamless linking with SZIP libraries when an encoding license is not available (see HDF4 and HDF5 “Software distribution” sections in “SZIP Compression in HDF Products”,
http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/

[1] It is important to note that the problems were in the integration of SZIP support in the HDF4 library. The SZIP library did not have significant problems, and the HDF5 integration of SZIP does not have problems.

Currently the HDF4 library is compiled into two libraries, libdf.a and libmfdf.a. These two libraries contain both C and Fortran HDF APIs. This prevents compilers from creating a shared library.

To fix the problem we have to separate the C and Fortran APIs and create new pure C and Fortran HDF4 libraries, i.e., one with C APIs and one with Fortran APIs. To avoid doubling the number of libraries with the old name schemas (libdf, libdf_fortran, libmfhdf, libmfhdf_fortran), we would like to combine all the C functions from libdf and libmfhdf in a single libhdf4 library and all the Fortran functions from libdf and libmfhdf in a single libhdf4_fortran library.

Changing the names of the libraries will affect make files and other compilations for any program that uses the HDF4 library. The source code will not need to change, just the build procedures.

The HDF-EOS2 library, the PGS toolkit, and all other software that uses HDF-EOS2 will need to change the names of the HDF4 libraries that they link to.

3. HDF-EOS5 / HDF5-1.8.0 issues

HDF5-1.8.0 will be released in July 2005. This revision includes many important changes, including major performance improvements, and new features.

Two changes in HDF5-1.8 will require minor changes to the HDF-EOS5 library. These are described in this section.

Early releases (numbers HDF5-1.7.xxx) are available for testing at:
ftp://hdf.ncsa.uiuc.edu/pub/outgoing/hdf5/snapshots

3.1. Error Reporting

HDF5-1.8.0 has a revised mechanism for returning errors from the HDF5 library. The new API is source-code incompatible with HDF-EOS. The problem is relatively minor: he HDF-EOS library uses the symbol 'HE5_NONE_MAJOR', which does not exist in HDF5-1.7 or later. (See Figure 1)

The recommended fix is to use the new H5E API to create error codes for HDF-EOS errors. These new functions will let the HDF-EOS library create its own error messages. For an example, see:
http://hdf.ncsa.uiuc.edu/RFC/ARCHIVE/ErrorAPI/Examples/Register.html

The new H5E API functions can be found at
http://hdf.ncsa.uiuc.edu/HDF5/doc_dev_snapshot/H5_dev/html/RM_H5E.html

For more information about new H5E APIs (a general discussion, use cases, usage examples) see:
http://hdf.ncsa.uiuc.edu/RFC/ARCHIVE/ErrorAPI/

```
  /* Check for valid HDFEOS file ID range */
  /* ------------------------------------ */
  if (fid < HE5_EHIDOFFSET || fid > HE5_NEOSHDF + HE5_EHIDOFFSET)
    {
        status = FAIL;
        sprintf(errbuf,"Invalid file ID: %d. ID should range from %d to  %d
.\n", fid, HE5_EHIDOFFSET, HE5_NEOSHDF + HE5_EHIDOFFSET);
        H5Epush(__FILE__, "HE5_EHchkfid", __LINE__, H5E_NONE_MAJOR,
H5E_BADVALUE, errbuf);
        HE5_EHprint(errbuf, __FILE__, __LINE__);
    }
  else
    {
```

Figure 1. Example of how H5E_NONE_MAJOR is used. (There are more than 300 instances of this symbol in the current release of HDF-EOS5.)

3.2. API changes: *hssize_t* is now *hsize_t* throughout HDF5 API

Several API functions were changed so that arguments are now type *hsize_t*. This change makes the variable unsigned, as it should been in the first place. The change was necessary to enhance performance of the HDF5 Library internals.

This change actually simplifies the HDF-EOS library code, which has a type cast just to work around the old declaration. The HDF-EOS5 library has two calls to *H5Sselect_elements* (in PTapi.c). These calls should be changed to pass the last argument as *hsize_t*.

herr_t `H5Sselect_elements`(*hid_t* `space_id`, *H5S_seloper_t* `op`,
const size_t `num_elements`, *const hssize_t* **`coord[ ]`)

should be changed to:

herr_t `H5Sselect_elements`(*hid_t* `space_id`, *H5S_seloper_t* `op`,
const size_t `num_elements`, *const hsize_t* *`coord[ ]`)

For the complete list of changed APIs see:
http://hdf.ncsa.uiuc.edu/HDF5/doc/ADGuide/Changes.html

4. Feature Request: HDF5 call to find all the 'aliases' for a field

We received a request for a new function that retrieves all the aliases for a field. HDF-EOS5 stores fields as HDF5 datasets, and creates "aliases" by creating additional links to the dataset. The HDF-EOS5 library has functions to create, remove, and look up these aliases.

Therefore, this request is asking for an HDF5 call that returns all the incoming links to an object. This call would enable programs to discover the alternative names from the HDF5 file without recourse to the HDF-EOS5 functions.

HDF5 does not have any mechanism for discovering all of the links to an object, except by traversing the entire group hierarchy. With the current library, it is possible to provide a routine (probably in the high level library) that searches the file to discover these links. This will be quite inefficient if there are a lot of objects in the file or if it is called many times.

Other solutions will require more extensive development, e.g., to change the storage format or add a feature to create and update a global table of links.

In short, there is no easy response for this request, and it will not be addressed in the next release. If this feature is a high priority, it will be necessary to scope out possible implementations in detail.