RFC: Options to handle compatibility issues for HDF5 files

Vailin Choi

The HDF5 Library 1.10.* introduces changes to the HDF5 File Format to improve performance for writing and reading chunked datasets, and to enable some new features such as SWMR. Applications that take advantage of those new features will create files that cannot be read by the applications built with HDF5 1.8. While the existing h5repack tool can be used to rewrite the file using earlier versions of the HDF5 File Format, it may be not an option when the file size if big (100MBs to TBs).

To mitigate the issue, we propose a tool to modify an HDF5 file in a way that will allow HDF5 1.8-based applications to read chunked data created with the 1.10 library without rewriting the whole HDF5 file. In the future the tool can be extended to provide data compatibility between any two versions of the HDF5 File Format.

Introduction

The 1.10 and 1.8 HDF5 libraries differ in the way chunked datasets are indexed. For a 1.10 HDF5 file generated with the latest file format, the 1.10 library uses one of the following indexing types depending on a chunked dataset's dimension specification and the way it is extended:

- Extensible array indexing for appending along a specified dimension
- Version 2 b-tree indexing for appending along multiple dimensions
- Fixed array indexing for fixed-size datasets
- Implicit indexing for 1-dim datasets (?)

The dataset layout information in the object header is described by a pair of messages:

- Version 4 *layout* message
- Version 0 *storage* message

The 1.8 library only supports version 1 b-tree indexing type for chunked datasets. The dataset layout information in the object header is described by a single message: version 1, 2, or 3 *layout* message.

We refer the reader to the latest File Format Specification [1] for description of the message and layout of the new chunk indexing.

The incompatibility described above will disallow 1.8 library-based applications to read a chunked dataset in a 1.10 HDF5 file generated with the latest library format.

This RFC discusses various approaches to a tool to "recreate" an HDF5 file in 1.8. File Format without rewriting raw data.

Option 1

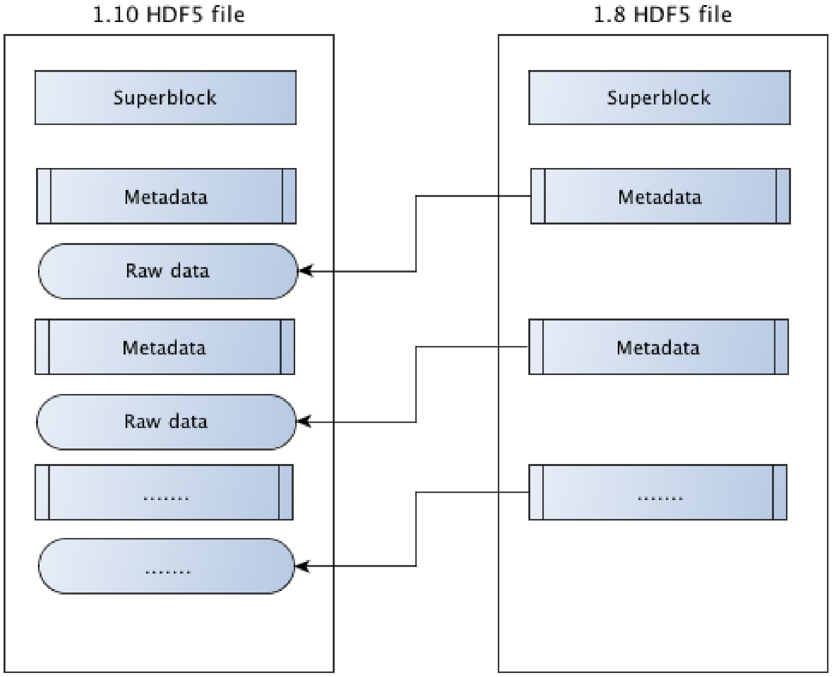
Precondition

User creates an HDF5 file with the 1.10 library's default sec2 driver, using the latest format. This is a case for files created for SWMR access.

The tool

It will create a light-weighted HDF5 wrapper file, which contains the following original grouping structure and datasets

- Datasets will have
  - HDF5 metadata from the 1.10 HDF5 file rewritten according to the 1.8. File Format (i.e., using B-tree version 1 chunk indexing).
  - Index will point to raw data chunks stored in the original1.10. HDF5 file



Pros

- The original 1.10 HDF5 file is not modified.
- A generic approach to resolve incompatibility.

Cons

- The HDF5 library does not currently support external storage for chunked layout; to do so will incur format changes for 1.8 library users. Thus this approach is not appropriate for 1.8 vs. 1.10 compatibility, but can be used starting with version 1.10 and later by introducing this feature in the HDF5 1.10.0.

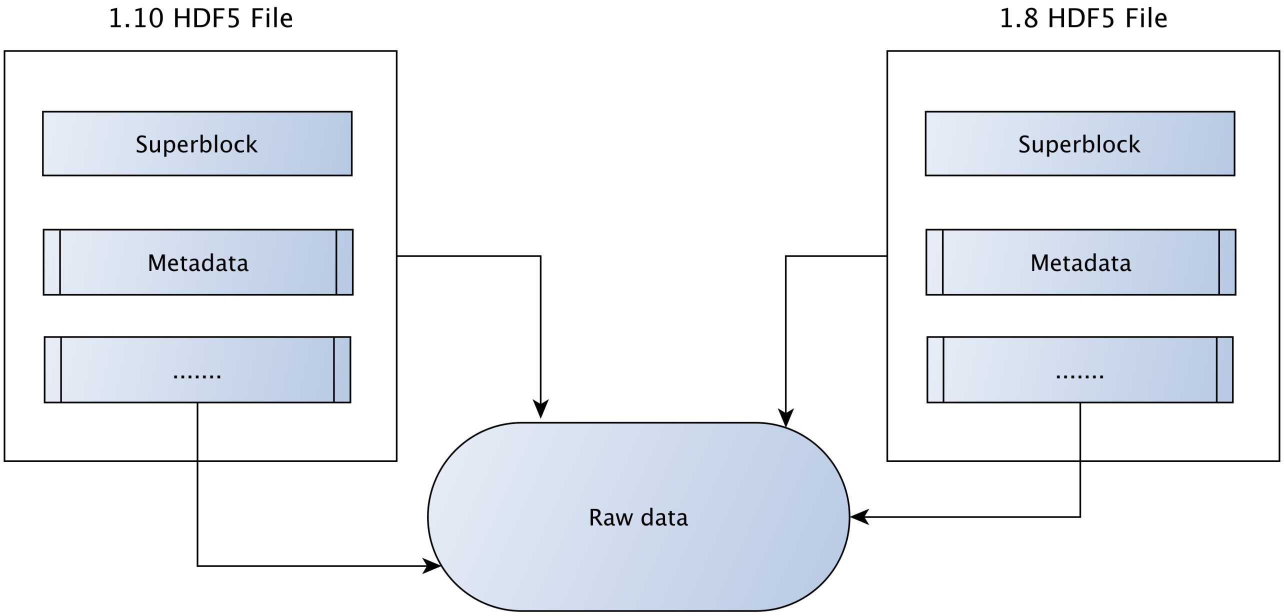
Option 2

Precondition

User creates an HDF5 file with the 1.10 library's split driver, using the latest format. This will result in two physical files: a metadata file and a data file.

The tool

It will create a 1.8 metadata file based on 1.10 metadata. User of the 1.8 library can read the chunked dataset with the split driver—using the 1.8 metadata file and the 1.10 data file. Both the 1.8 and 1.10 libraries will be modified to detect the file driver employed on file creation, and to use such driver on subsequent file opens.



Pros

- The original 1.10 HDF5 file (the split file) is not modified.
- A generic approach to resolve incompatibility.

Cons

- Users will have two physical files and have to keep them together to access the data.
- The HDF5 file has to be rewritten to get one physical file.
- The HDF5 1.8. Library has to be modified to automatically detect the split file driver. The change can be done in the HDF5 1.8.14 release (November 2014). Applications that cannot switch to HDF5 1.8.14 will not be able to read such files without modifications to detect a split

file.

Option 3

Precondition

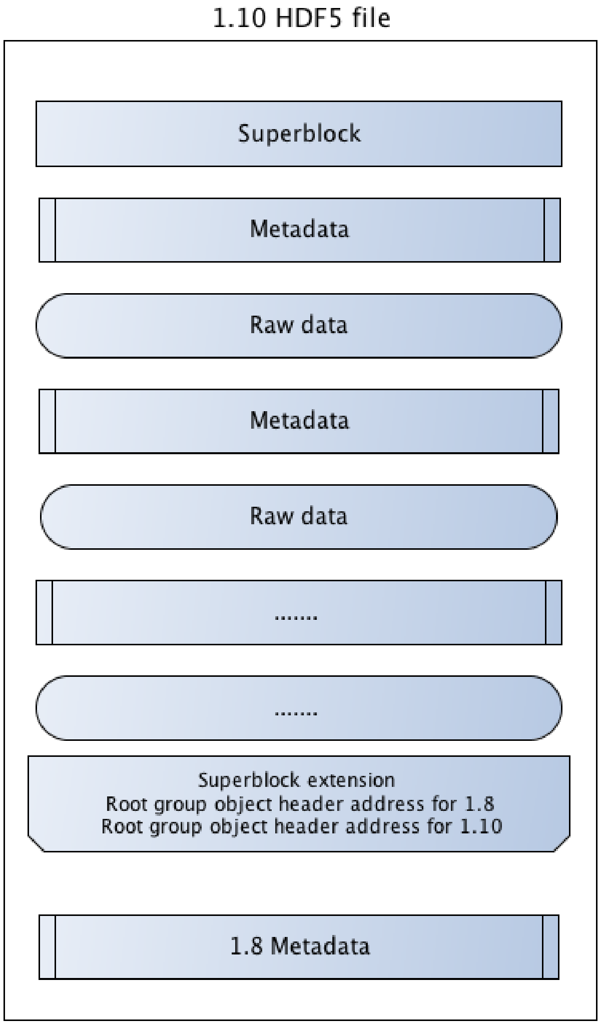
User creates an HDF5 file with the 1.10 library's default sec2 driver, using the latest format.

The tool

It will create and append the following information to the 1.10 HDF5 file:

- Duplicated 1.10 metadata rewritten according to the 1.8 File Format.
- A new message in the superblock extension's object header that contains:
    - Root group object header address for 1.8
    - Root group object header address for 1.10

Information in the new message enables a user to switch between 1.10 and 1.8 metadata via an option in the tool.



Pros

- A generic approach to resolve incompatibility.
- Flexibility to switch between 1.8 and 1.10 metadata in a single HDF5 file.
- No modifications to the HDF5 1.8 library since it can ignore unknown messages.

Cons

- The original 1.10 HDF5 file is modified.
- Application has to specify the file format version and link with appropriate library to read data.
- If application modifies a file according to one version of the files format, metadata corresponding to another file format has to be updated, otherwise the new data will not be accessible to application based on the second version of the file format.
- Current HDF5 1.10 library has to be modified to choose an appropriate version.

Option 4

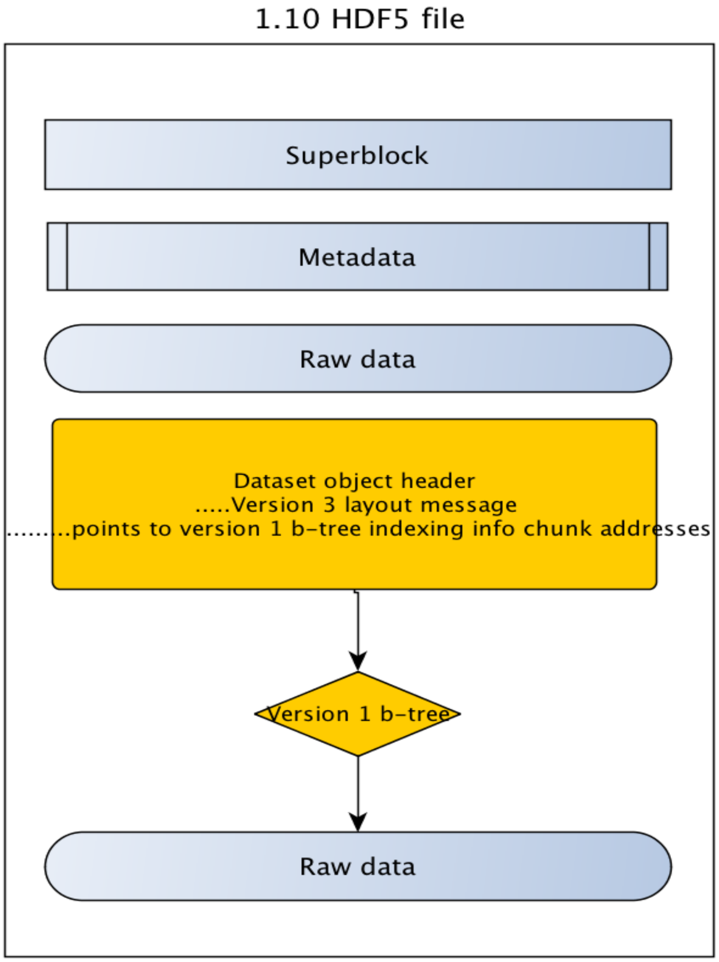
Precondition

User creates an HDF5 file with the 1.10 library's default sec2 driver, using the latest format.

The tool

It will convert the latest indexing type of a specified chunked dataset to version 1 b-tree indexing type. The tool will traverse the object header of the dataset to perform the following:

- Create version 1 b-tree indexing for the dataset.
- Create version 3 *layout* message to point to the version 1 b-tree.
- Retrieve the set of chunk addresses *K* from the indexing information in the *storage* message
- Insert the set of chunk addresses *K* into the version 1 b-tree.
- Delete the *storage* and the version 4 *layout* messages from the object header without deleting file space for the chunks.


1.10 HDF5 file
Superblock
Metadata
Raw data
Dataset object header
.....Version 3 layout message
........points to version 1 b–tree indexing info chunk addresses
Version 1 b–tree
Raw data

Pros

- A specific approach to resolve chunk-indexing incompatibility.
- A quick way to allow 1.8 library users to read a specific chunked dataset in the 1.10 HDF5 file.

Cons

- The 1.10 HDF5 file will be modified intensively.
- Information of the dataset's latest indexing type will be lost.
- Cannot simultaneously use the 1.8 and 1.10 libraries with the file.

Option 5

Precondition

User creates an HDF5 file with the 1.10 library's default sec2 driver, using the latest format.

The tool

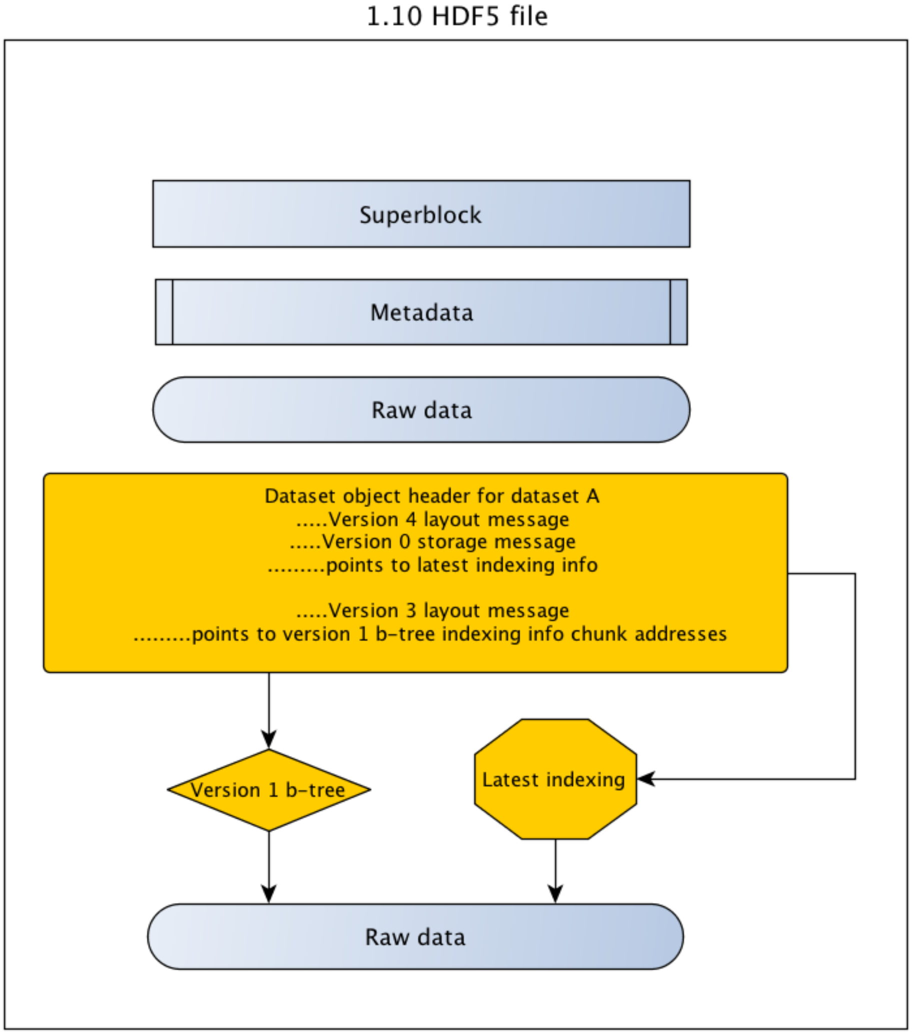
This is a variation of the tool described in Option 4. It will traverse the object header of the specified chunked dataset to perform the following:

- Create version 1 b-tree indexing for the dataset.
- Create version 3 *layout* message to point to the version 1 b-tree.
- Retrieve the set of chunk addresses *K* from the indexing information in the *storage* message.
- Insert the set of chunk addresses *K* into the version 1 b-tree.

For this option, the object header of the specified chunked dataset will have dual dataset layout information as below:

- The newly created version 3 *layout* message indicating version 1 b-tree indexing type.
- The version 4 *layout* message and the *storage* message indicating the latest indexing type.

Both the 1.8 and 1.10 libraries will be modified to detect and apply the appropriate dataset layout information. Areas in the library that manipulate object header messages will also be modified to properly handle such dual layout information.


1.10 HDF5 file
Superblock
Metadata
Raw data
Dataset object header for dataset A
.....Version 4 layout message
.....Version 0 storage message
.........points to latest indexing info
.....Version 3 layout message
.........points to version 1 b-tree indexing info chunk addresses
Version 1 b-tree
Latest indexing
Raw data

Pros

- A specific approach to resolve chunk-indexing incompatibility.
- Flexibility to use either indexing types for a chunked dataset depending on the library in use.

Cons

- The 1.10 HDF5 file will be modified intensively.
- The HDF5 1.8 library has to be modified to detect such situation.
- Applications will need to switch to the updated HDF5 1.8 library version.

Option 6

Precondition

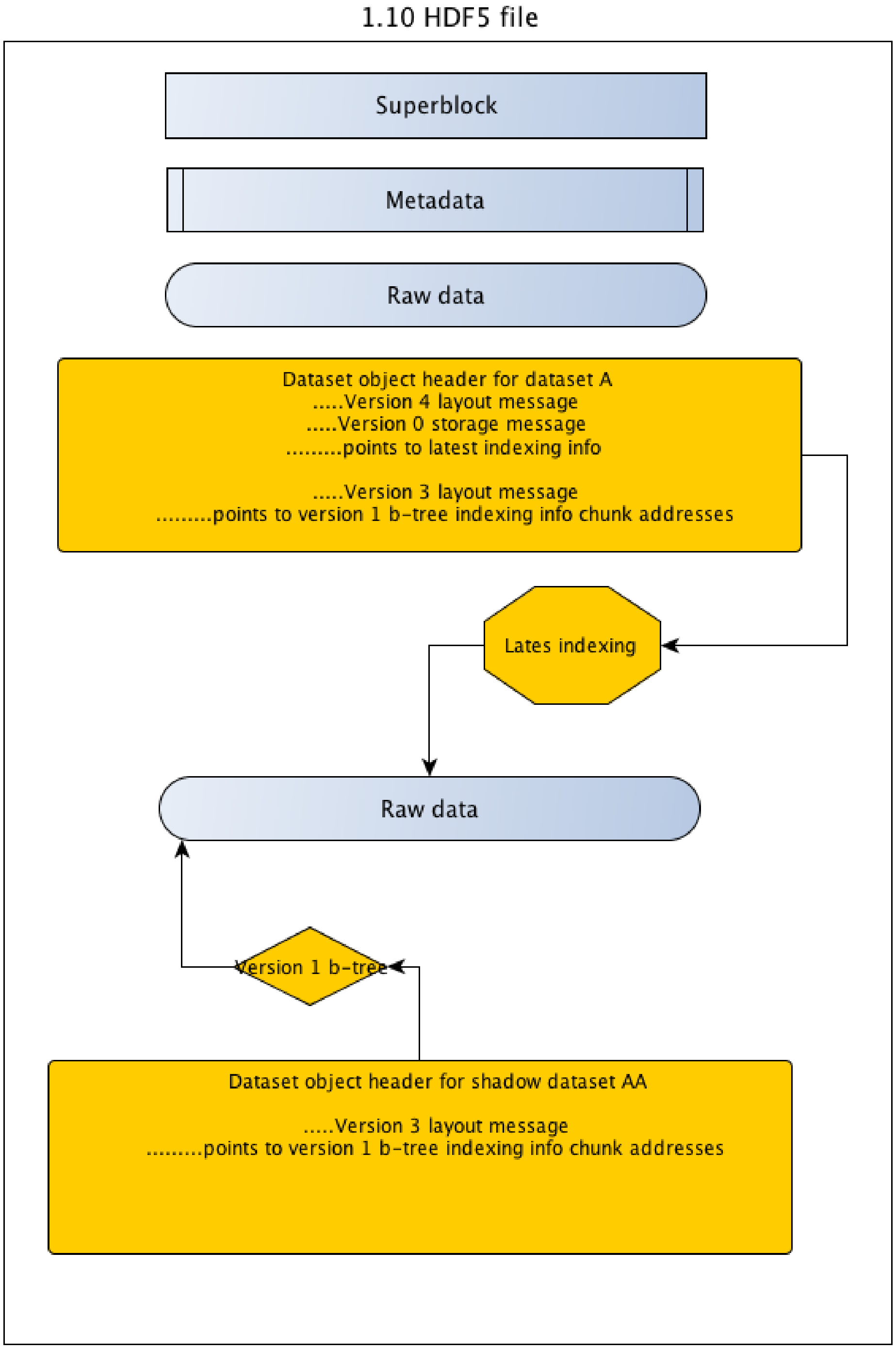
User creates an HDF5 file with the 1.10 library's default sec2 driver, using the latest format.

The tool

The existing library tool *h5copy* will be used to copy a specified chunked dataset *A* to a shadow dataset *AA* in the 1.0 HDF5 file. There will be a new object copy flag that *h5copy* can pass to the public routine *H5Ocopy*, indicating shadow copy of a chunked dataset that 1.8 library can understand. The copy process in *H5Ocopy* will be modified as below:

- To allow the copying of source object to destination object with different indexing types.
- To copy the set of chunk addresses *K* instead of allocating new file space for the chunks in the destination dataset.

This option will result in two datasets: *A* and *AA* both point to the same set of chunk addresses *K*. Both libraries will need to have proper handling on deletion of such datasets.


1.10 HDF5 file
Superblock
Metadata
Raw data
Dataset object header for dataset A
.....Version 4 layout message
.....Version 0 storage message
.........points to latest indexing info
.....Version 3 layout message
.........points to version 1 b-tree indexing info chunk addresses
Lates indexing
Raw data
Version 1 b-tree
Dataset object header for shadow dataset AA
.....Version 3 layout message
.........points to version 1 b-tree indexing info chunk addresses

Pros

- A specific approach to resolve chunk-indexing incompatibility.
- The dataset with the latest indexing type is kept intact.
- Can make use of existing library tool and public routine.

Cons

- The original 1.10 HDF5 file will be modified
- There will be a shadow dataset for each chunked dataset being copied.

Conclusion

The six options described in this RFC can be categorized into two groups: the first three options are more general in nature, while the last three options are more specific. The selection to make will depend on time and cost factors. After choosing the desired option, we might also consider the appropriate implementation approach: as a tool built in the 1.10 library or as an independent tool like *h5check*.

Acknowledgements

This design work is supported by The HDF Group's internal Sustaining Engineering project (former GMQS).

Revision History

March 13, 2014:	Version 1 circulated for comment within The HDF Group.
March 20, 2014	Version 2 included comments from the team members and was slightly reformatted. MS Word file was renamed to reflect the subject of this RFC; sent to the group.

References

1. HDF5 File Format Specifications http://svn.hdfgroup.uiuc.edu/hdf5doc/branches/revise_chunks/html/H5.format.html
2. New Features in HDF5 Release 1.8.0 and Format Compatibility Considerations http://www.hdfgroup.org/HDF5/doc/ADGuide/CompatFormat180.html