RFC: Generating attribute into an object with a tool

Jonathan Kim

This RFC proposes four suggestions to generate attribute(s) with a command-line tool.

Each approach needs to be reviewed and discussed for reaching a final decision.

1 Introduction

A feature to generate attribute(s) into an object with a tool was requested by NPOESS project.

The four suggestions are

- First - Integrating the feature into h5import tool.
- Second – As a new tool, following user interface from the h5import.
- Third – As a new tool, getting required input from command arguments.
- Fourth – As a new tool, getting required input from DDL format file.

This RFC was prepared initially with the first suggestion in mind and the other suggestions were added from feedback. More details will be provided in another RFC based on the final decision.

2 Review for required input for generating an attribute

Data types

- Simple input class
  - H5T_INTEGER
  - H5T_FLOAT
  - H5T_STRING
- Complex input class
  - H5T_ENUM
  - H5T_COMPOUND
  - H5T_VLEN
  - H5T_ARRAY
  - H5T_OPAQUE
  - H5T_BITFIELD
  - H5T_REFERENCE

Data space

- RANK
- DIMS

Target object

- Path to an existing object

Attribute name

- A string name for generating an attribute

Data values

- Data values according to the data type

3 Suggestion 1: Integrating the feature into h5import tool

Currently the 'h5import' tool can create or add dataset(s) into a HDF5 file with limited data types. The supported data types are 'Integer', 'Floating point' and 'String'. Other new data types will need to be supported as necessary. (ex: reference type, …)

Since a dataset and an attribute in HDF5 are very similar to each other, the 'h5import' tool can be used to generate attribute(s) to object(s) in a similar manner. So users who are already familiar with the h5import tool will easily be able to adapt to generate attribute(s) by using the same h5import tool.

For the implementation, certain code in the h5import can be reused along with the current frame work. Since the dataset and the attribute are similar to each other, future additional code for other data types can be shared between the dataset and attribute.

Please refer to the appendix for more details.

4 Implement as a new tool

3 suggestions are included in this section.

This proposal would provide flexibility to be evolved and maintained independently from the purpose of h5import tool. Also it could give the user a tool which only focuses on the attribute more intuitively.

For the implementation, the ***H5LTtext_to_dtype*** function is considered to be used for handling various data types.

Internal feedback for a new tool approach

- Distinguish from the purpose of h5import tool.   "The h5import is designed for importing large existing data file into HDF5 datasets."
- Generating attribute(s) with command-line parameters (using the text-to-datatype feature) for simple structure quickly.
- Use HDF5 DDL format file (output from h5dump) for required input and evolve for generating other objects and using input format like XML.

4.1 Suggestion 2: Following user interface from the h5import

This will provide uniform user interface from the h5import tool for the user.

- Input for generating attribute
  - As a configuration file.
  - Specify the file with '–c' command line argument
- Input for data values of the attribute
  - As a text based file with data values.
  - Specify the file without a particular command line argument

4.2 Suggestion 3: Getting required input from command line arguments

This method can provide capability to generate simple attributes quickly via command line arguments. However handling more complex data types and data values would not be practical since it could be too lengthy and complex.

- Example for simple attribute
  - <CMD> --type H5T_NATIVE_INT--space 1,3 --obj /dset1 --name attr_levels --data 10,20,30
  - <CMD> --type H5T_STD_I8BE--space 1,3 --obj /dset1 --name attr_levels --data 10,20,30
- Need to discuss how to handle complex data types and values :
  - Example for complex data type input
    - H5T_STRING { STRSIZE H5T_VARIABLE; STRPAD H5T_STR_NULLPAD; CSET H5T_CSET_ASCII; CTYPE H5T_C_S1; }
    - H5T_ENUM { H5T_STD_I32LE; \"RED\" 5; \"GREEN\" 6; \"BLUE\" 7; \"WHITE\" 8; }
    - H5T_VLEN { H5T_VLEN { H5T_STD_I32BE } }
    - H5T_COMPOUND { H5T_STD_I16BE \"one_field\" : 2; H5T_STD_U8LE \"two_field\" : 6; }
    - H5T_OPAQUE { OPQ_SIZE 19; OPQ_TAG \"This is a tag for opaque type\”
    - H5T_ARRAY { [5][7][13] H5T_ARRAY { [17][19] H5T_COMPOUND { H5T_STD_I8BE \"arr_compound_1\"; H5T_STD_I32BE \"arr_compound_2\"; } } }
    - Reference type not implemented
  - Data values can be complex as well.

4.3 Suggestion 4: Getting required input from DDL format file

HDF5 DDL format includes all the information required to generate an attribute. The DDL format is human readable and it can be easily edited by hand. The DDL output can be easily obtained from h5dump tools and can be reused as a template.

We can also provide sample template files or HDF5 files with various complexities, so any user can easily edit the files to generate attribute(s).

If this plan becomes successful, the same concept can be applied to generate other objects. Eventually the combination of all the common functions will provide capability to generate a HDF5 file with object(s) and attribute(s) from a DDL file input.

In the long run, the code framework can be designed in a way that the XML format file can be used as input as well.

- **Example of integer type**

```
ATTRIBUTE "A1" {
      DATATYPE  H5T_STD_I64BE
      DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
      DATA {
      (0,0): 0, 1, 2, 3, 4
      }
    }
```

- **Example of string type**

```
ATTRIBUTE "VERSION" {
      DATATYPE  H5T_STRING {
          STRSIZE 4;
          STRPAD H5T_STR_NULLTERM;
          CSET H5T_CSET_ASCII;
          CTYPE H5T_C_S1;
        }
      DATASPACE  SCALAR
      DATA {
      (0): "2.0"
      }
    }
```

- **Example of complex type**

```
ATTRIBUTE "REFERENCE_LIST" {
      DATATYPE  H5T_COMPOUND {
        H5T_REFERENCE "dataset";
        H5T_STD_I32LE "dimension";
      }
      DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
      DATA {
      (0): {
          DATASET 800 /dset_al ,
          3
        },
      (1): {
          DATASET 4504 /dset_al2 ,
          3
        }
      }
    }
```

5 Considerations

- Is this a feature for a particular customer or for general users?
- When is the deadline?
- Is it possible to implement before the deadline?
- Which suggestion would benefit the company most for the long term?
- Open to feedback?

6 Appendix

6.1 Example to create a dataset with h5import tool

Command line

- h5import **txtin16.txt** **–c** **txtin16.conf** **-o** txtin16.h5

Configurations file '**txtin16.conf**'

```
PATH /int/16-bit
INPUT-CLASS TEXTIN
INPUT-SIZE      16
RANK 3
DIMENSION-SIZES 2 4 3
OUTPUT-BYTE-ORDER LE
CHUNKED-DIMENSION-SIZES 2 2 2
MAXIMUM-DIMENSIONS -1 -1 8
OUTPUT-ARCHITECTURE STD
```

Data values file – '**txtin16.txt**'

```
83      85      87      89
84      86      88      90
85      87      89      91
88      90      92      94
89      91      93      95
90      92      94      96
93      95      97      99
94      96      98      100
95      97      99      101
98      100     102     104
99      101     103     105
100     102     104     106
103     105     107     109
104     106     108     110
105     107     109     111
```

Result from h5dump

```
HDF5 "txtin16.h5" {
GROUP "/" {
   GROUP "int" {
      DATASET "16-bit" {
         DATATYPE  H5T_STD_I16LE
         DATASPACE  SIMPLE { ( 2, 4, 3 ) / ( H5S_UNLIMITED, H5S_UNLIMITED, 8 ) }
         DATA {
         (0,0,0): 83, 85, 87,
         (0,1,0): 89, 84, 86,
         (0,2,0): 88, 90, 85,
         (0,3,0): 87, 89, 91,
         (1,0,0): 88, 90, 92,
         (1,1,0): 94, 89, 91,
         (1,2,0): 93, 95, 90,
         (1,3,0): 92, 94, 96
         }
      }
   }
}
}
```

6.2 Details for the first suggestion

6.2.1 Configuration directives

The h5import tool uses configuration file as input to generate a dataset. Portion of the current directives can be reused for generating an attribute in the same manner.

6.2.1.1 Current directives for creating 'Dataset' from h5import

- PATH
- INPUT-CLASS
- INPUT-SIZE
- RANK
- DIMENSION-SIZES
- OUTPUT-CLASS
- OUTPUT-SIZE
- OUTPUT-ARCHITECTURE
- OUTPUT-BYTE-ORDER
- CHUNKED-DIMENSION-SIZES
- MAXIMUM-DIMENSIONS
- COMPRESSION-TYPE
- COMPRESSION-PARAM
- EXTERNAL-STORAGE

6.2.1.2 Directives which can be reused for creating 'Attribute'

- PATH
  - A path to an attribute. This should contain a valid path to an existing object.
  - Ex: /group1/dset100/attr_test
    - /group1/dset100 must be an existing object in a given HDF5 file.
- INPUT-CLASS
- INPUT-SIZE
- RANK
- DIMENSION-SIZES
- OUTPUT-CLASS
- OUTPUT-SIZE
- OUTPUT-ARCHITECTURE
- OUTPUT-BYTE-ORDER

Note: directives not supported for attribute

- CHUNKED-DIMENSION-SIZES
- MAXIMUM-DIMENSIONS
- COMPRESSION-TYPE
- COMPRESSION-PARAM
- EXTERNAL-STORAGE

6.3 Useful High-Level library functions

6.3.1 Getting various data types from a text input

- H5LTtext_to_dtype
  - Reference type not yet supported

6.3.2 Generating attribute with simple data types

- H5LTset_attribute_string
- H5LTset_attribute_char
- H5LTset_attribute_uchar
- H5LTset_attribute_short

- H5LTset_attribute_ushort
- H5LTset_attribute_int
- H5LTset_attribute_uint
- H5LTset_attribute_long
- H5LTset_attribute_long_long
- H5LTset_attribute_ulong
- H5LTset_attribute_float
- H5LTset_attribute_double

Revision History

April 22, 2010:	Version 1 draft for initial review.
April 30, 2010:	Version 2 revised for internal review