

Proposal for representing simple data in the HDF5 XML DTD

Mike Folk, May 23, 2000

Rev 2: Revised May 31, 2000

Rev 3: Revised June 8, 2000

Background

For reasons given in the design notes (section 3.3), we were not able to create a satisfactory markup for data in the HDF5 file for the first release. The initial version of the DTD has a limited <DataFromFile> element, which does not support all the desired features. This will be revised in a future release, after we have had more time and experience with this the HDF5 DTD.

So the purpose of this proposal is just to suggest one way to represent data in HDF5, not to provide a general solution for representing HDF5 data. Our purpose here is to help users experiment with the DTD in useful ways, and to get feedback on this and other possible approaches.

Comments, corrections, and suggests are very welcome.

Data ordering

In the proposed scheme, values shall listed in "row major" order, meaning that the last dimension varies fastest. For instance, for a three dimensional array, 4 x 5 x 10, declared as

```
int x[4][5][10]
```

the data would appear in the order:

```
x[0][0][0] x[0][0][1] ... x[0][0][9] x[0][1][0] x[0][1][1] ... x[0][1][9] ...  
x[3][4][0] x[3][4][1] ... x[3][4][9]
```

Datatypes

All HDF5 datatypes are covered except opaque and object selection references. Any compound datatype that includes either of these cannot be displayed. Individual values must be represented by one of the following three types: decimal integer constant, character string, or floating point constant. Datatypes are mapped to these three types as follows:

Datatypes	Representation	Comments/examples
All integers	<u>integer_const</u>	Decimal integers only
All floats	<u>float_const</u>	
Characters and strings	<u>string_const</u>	Use C rules for special characters
Enum	<u>string_const</u>	
Bitfield	<u>binary_const</u>	Assumed to be 0s and 1s.
Object reference	<u>integer_const</u>	Assumes no decimal point
Variable length type*	<u>paren_list</u>	Except string. E.g. (4,5,6)
Compound	<u>compound_list</u>	E.g. 5.2, "abc" or [5.2, "abc"]

*Except character string

BNF notation for <DataFromFile> element

This BNF description of a valid <DataFromFile> stream uses the following special notation:

<token>?: 0 or 1 occurrences of <token>
<token>*: 0 or more occurrences <token>
<token>+: 1 or more occurrences <token>

```
digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
non_zero_digit ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
integer_const ::= sign non_zero_digit digit* // assume decimal integer
float_const ::= sign fraction exp part? |
                 integer_const exp part
fraction ::= digit* "." digit*
exp part ::= ("e" | "E") sign digit*
sign ::= "+" | "-" | <epsilon> // <epsilon> indicates the
                                // absence of a token
binary_const ::= ("0" | "1")+

valid_char ::= // Any character accepted by HDF5 (except "<"
                // and "&") for the datatypes that are supported
                // "&lt;" must be used instead of "<" and "&"
                // instead of "&"
string_const ::= quote valid_char* quote // allows empty strings?
quote ::= "\"" // the double-quote symbol
paren_list ::= "(" list* ")" // allows empty variable-
                                // length types
compound_list ::= "{" list* "}" | list*
list ::= value (separator value)* separator? / ws*
value ::= integer_const |
            float_const |
            string_const |
            paren_list |
            compound_list
separator ::= ws+ | // allow any combination of separators
                ws* "," ws* // but no more than one comma
ws ::= <space> | // white-space
        <tab> |
        <newline> |
        <carriage_return>
```