RFC: Refactored h5dump Program Flow and Process

Allen Byrne

This document details how the h5dump program and the tools library work, based on the code in the tools\_refactor feature branch of the hdf5 library.

# Introduction

The h5dump program processes an hdf5 file based on parameters passed on the command line. By default, the main function starts with the root group and iterates through all the objects. Each object is parsed into blocks and these blocks are then displayed. This document describes the process.

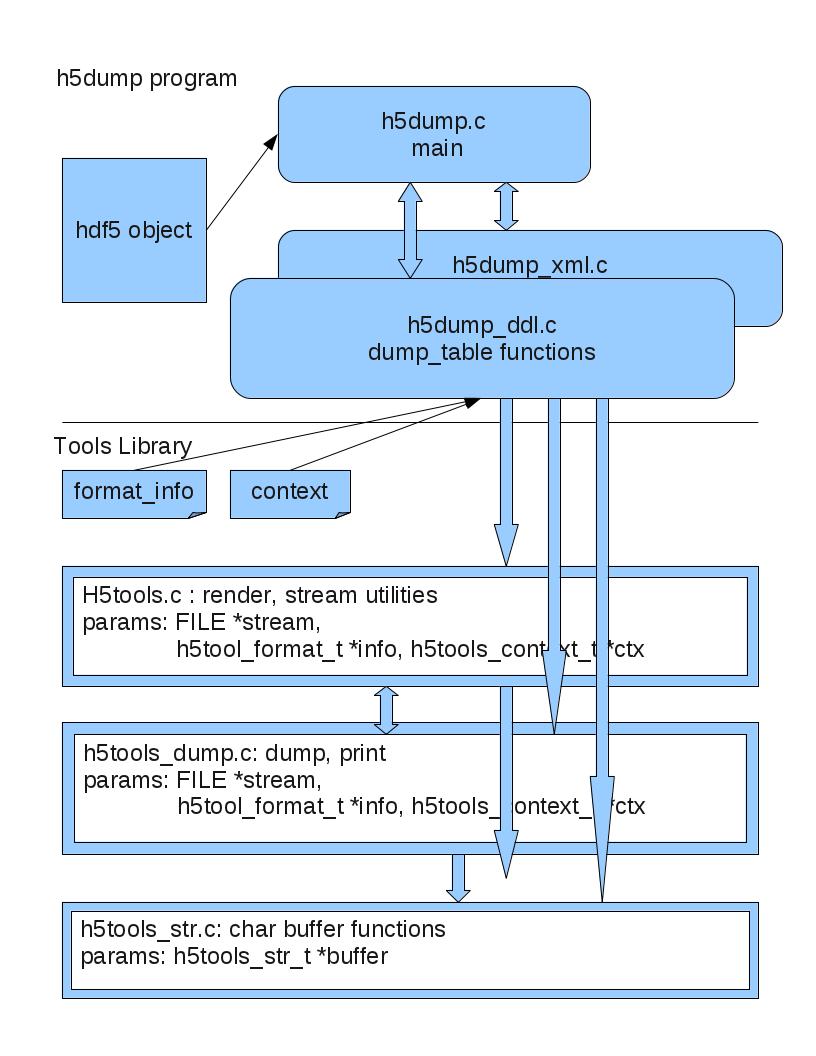
# h5dump program flow

After parsing the command line, the main function initializes some program variables and begins the walk through the hdf5 file’s objects. By default, the main function starts with the root group and iterates through all the objects. Iteration is controlled by the callback function table, **dump\_function\_table**, and the h5dump program either calls a function in the table directly or within an hdf5 library iterates call.

Each high-level function of the **dump\_function\_table** creates and initializes the independent format and context structures using the global variable **dump\_indent** to configure the context indentation. The context structure controls output interaction with the **dump\_function\_table** functions by aligning the **context->indent** variable and the global **dump\_indent** variable. The formatting structure is rarely changed from the default initialization.

The h5dump program files concentrate on the functional control flow and the tools library files concentrate on the object formatting. The low-level **h5tools\_str** functions are used as building block functions to allow more general content dump functionality to be built into the **h5tools\_dump** functions, which concentrate on how to display the content.

Figure 1 Refactored Code Layout



The two structures used to control the output of information from h5dump; **h5tool\_format\_t** and **h5tools\_context\_t**, and the necessary #defines are located in the tools library (see **h5tools.h**). The format structure, **h5tool\_format\_t**, members are strings that define the printf() format string used to print the variables and the element markup. These are set to default values in the **h5dump** file functions and can be changed depending on the process.

Example of members of **h5tool\_format\_t** :

|  |
| --- |
| fmt\_double: The printf() format to use when rendering data which is  typed `double'. The default is `%g'. |
| Fields associated with the index values printed at the left edge of  each line of output.    n\_fmt: Each index value is printed according to this printf(3c)  format string which should include a format for a long  integer. The default is "%lu".    sep: Each integer in the index list will be separated from the  others by this string, which defaults to a comma.    fmt: After the index values are formated individually and  separated from one another by some string, the entire  resulting string will be formated according to this  printf(3c) format which should include a format for a  character string. The default is "%s". |

The global variable **dump\_indent** is initialized by the main function.

There is a general data output sequence followed by the dump functions in the h5dump program and the tools library:

Output an optional newline and prefix, followed by a reset of the output buffer. Then append text to the buffer, and conclude with the rendering of the buffer to the output stream.

This general sequence can be repeated as necessary, and any information block is usually contained within an element BEGIN/END pair that indicates the type of information and its bounds. These BEGIN/END pairs are controlled by the structure; **h5tools\_dump\_header\_t**. For example, the GROUP object would display as follows with two pairs (the second pair BLOCK BEGIN/END is the brackets):

|  |  |
| --- | --- |
| …  GROUP “groupname” {  …  }  … | “BEGIN ELEM” “name” “BLOCK BEGIN”  …  “BLOCK END” “END ELEM” |

The **h5tools\_dump\_header\_t** members used in the example above are:

const char \*groupbegin; //initialized to H5\_TOOLS\_GROUP

const char \*groupend; //initialized to “”

const char \*groupblockbegin; //initialized to “{“

const char \*groupblockend; //initialized to “}”

The **h5tools\_context\_t** structure (see Appendix for structure) is used to control where to place element rendering in a column defined output line or group of lines. The member **indent\_level** is manipulated most often and is initialized by the h5dump functions from the global variable; **dump\_indent**. The number of columns per indent level is controlled by the **h5tool\_format\_t** member; **line\_indent**.

The basic low-level operations in the **h5tools\_str** file operate only on an output buffer using a supplied output format variable. The **h5tools\_dump** file contains the functional operations involving the stream file and output format passed in from a high-level function in the h5dump program. Utility functions in **h5tools** and **h5tools\_utils** provide functions that support the interface between these two files, **h5tools\_str** and **h5tools\_dump**. The h5tools\_str.c file still provides string manipulation utility functions to **h5tools\_dump** by the IN/OUT variable: **buffer**.

# The h5dump Process

The h5dump program processes objects and generates three distinct output formats; binary data, standard DDL text and XML text. At this time XML output formatting is not addressed by the current refactoring effort (other than to synchronize any overlapping uses of functions in the tools library) and will be addressed in another document. The DDL format is described using BNF as documented at: <http://www.hdfgroup.org/HDF5/doc/ddl.html>. It remains to be determined if the XML formatting requirements can be incorporated with the standard DDL format process. The XML output is described by: <http://www.hdfgroup.org/HDF5/XML/DTD/HDF5-File.dtd>. The dump process starts with the h5dump main function and other high-level functions are called through the callback function table, **dump\_function\_table**, either directly or within an iterate call.

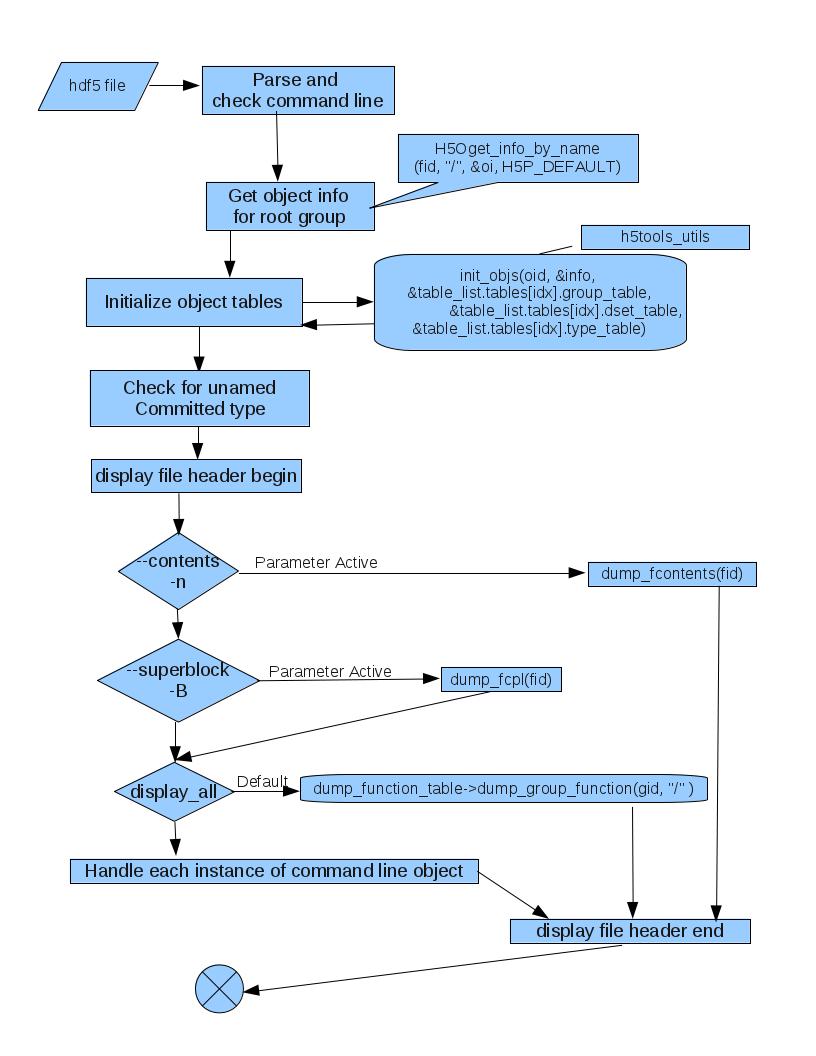
# DUMP\_FUNCTION\_TABLE CODE EXCERPTS

|  |
| --- |
| /\* Standard DDL output \*/  static const dump\_functions ddl\_function\_table = {  dump\_group,  dump\_named\_datatype,  dump\_dataset,  dump\_dataspace,  dump\_datatype,  dump\_attr\_cb,  dump\_data  };  …  /\* main function \*/  …  h5tools\_dump\_header\_format = &h5tools\_standardformat;  dump\_function\_table = &ddl\_function\_table;  dump\_indent = 0;  … |

## h5dump high-level functions

**The** **dump\_function\_table allows the future creation of public API interfaces to the dump/print functions of the tools library.**

### main function



The main function, after processing the command line, initializes the group, dataset, and type object tables. These tables keep track of already discovered objects during the dump process. An object is only added into the table when it is discovered. The first displayed object is the FILE header.

If the option to display the file contents has been selected, h5dump iterates through all the objects printing the type (group, dataset, datatype, or unknown) and name. There is no further processing after the data is displayed and the program exits.

# FILE CONTENTS EXAMPLE

|  |
| --- |
| > h5dump –n fcontents.h5  HDF5 "fcontents.h5" {  FILE\_CONTENTS {  datatype /#5616  group /  dataset /dset  dataset /dset3 -> /dset  dataset /dset4 -> /dset  dataset /dsetmytype2  ext link /extlink -> fname oname  group /g1  dataset /g1/dset1 -> /dset  group /g1/g1.1  dataset /g1/g1.1/dset2 -> /dset  group /g2 -> /g1/g1.1  link /mylink -> mylink  datatype /mytype  link /softlink -> /dset  unknown type of UD link /udlink -> ???  }  } |

**The content of the super block is printed if the option has been selected. The final h5dump option is either print “all” or print a single object (attribute, dataset, group, link, or datatype) per command line instance. The display all option is the default and is turned off by the use of any of the individual object options. Either display option uses the same dump\_function\_table functions for formatting the output. Dumping the individual group ”/” will output the same information as dumping by default the whole file.**

# ****SUPER BLOCK EXAMPLE****

|  |
| --- |
| >h5dump –B super\_block.h5  HDF5 "super\_block.h5" {  SUPER\_BLOCK {  SUPERBLOCK\_VERSION 0  FREELIST\_VERSION 0  SYMBOLTABLE\_VERSION 0  OBJECTHEADER\_VERSION 0  OFFSET\_SIZE 8  LENGTH\_SIZE 8  BTREE\_RANK 16  BTREE\_LEAF 4  ISTORE\_K 32  FILE\_SPACE\_STRATEGY H5F\_FILE\_SPACE\_ALL  FREE\_SPACE\_THRESHOLD 1  USER\_BLOCK {  USERBLOCK\_SIZE 0  }  }  GROUP "/" {  }  } |

### dump\_function\_table functions

The **dump\_function\_table** has seven functions with the dump\_XXX signature:

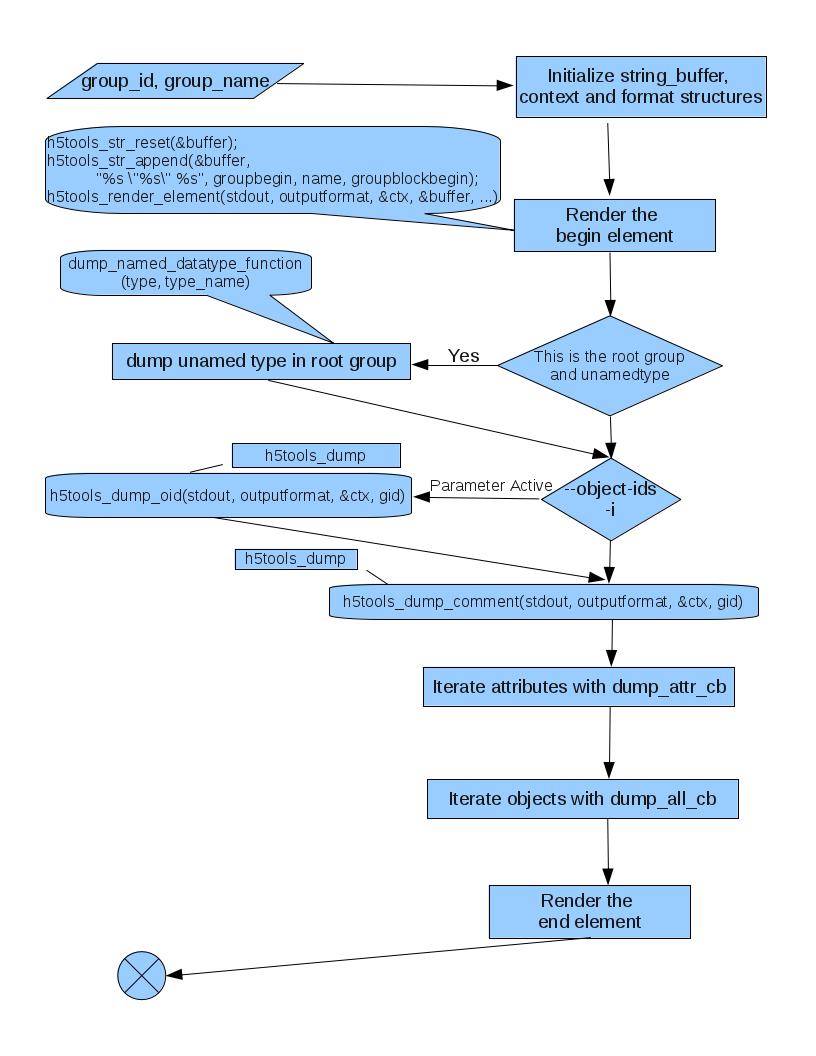
* group
* named datatype
* dataset
* dataspace
* datatype
* attr
* data

All **dump\_function\_table** functions are independent from each other and only global variables are accessible. Therefore, each **dump\_function\_table** function creates new format and context structures, which are used with all tools library function calls generated from the function. A global variable for indentation is used to communicate indentation location between **dump\_function\_table** functions.

# TOOLS INITIALIZATION EXAMPLE CODE

|  |
| --- |
| void dump\_xxxx(hid\_t id, const char \*name)  {  …  h5tools\_str\_t buffer; /\* string into which to render \*/  h5tools\_context\_t ctx; /\* print context \*/  h5tool\_format\_t \*outputformat = &h5tools\_dataformat;  …  /\* setup \*/  HDmemset(&buffer, 0, sizeof(h5tools\_str\_t));  memset(&ctx, 0, sizeof(ctx));  ctx.indent\_level = dump\_indent/COL;  ctx.cur\_column = dump\_indent;  … |

#### dump\_group



The **dump\_group** function is the default first function and after printing the name of the group it will output information about the group.

If this is the root group and there are any unamed datatypes, they will be printed first.

If the option to print the ObjectID is selected, it will be printed followed by any comment on the group before iterating through the objects contained within the group.

Before iterating through the attributes and other non-attribute objects of the group, the function checks if he current object has been displayed before or is an external link. If this is the case, only the object name is printed.

The final two processes in the **dump\_group** function iterate through the contained objects following the type of order – creation or name as specified on the command line. If there is a request to do H5\_INDEX\_CRT\_ORDER and tracking order is set in the group for attributes, then, sort by creation order, otherwise by name.

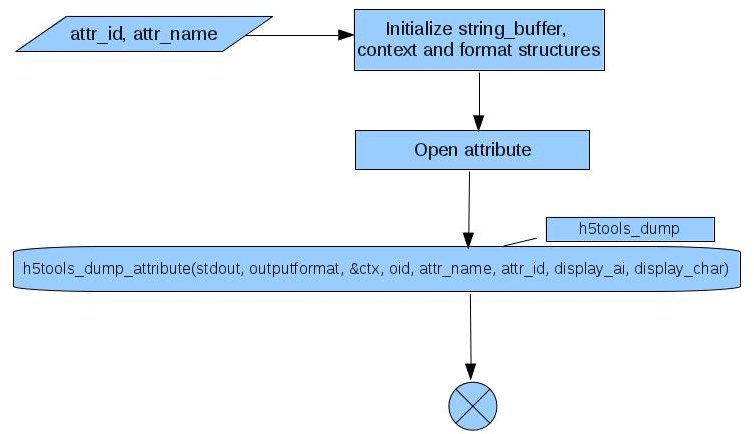
##### Iterate through the Attributes

The function **dump\_attr\_cb** is used as the callback in the H5Aiterate call. If there is a request to do H5\_INDEX\_CRT\_ORDER and tracking order is set in the group for attributes, then, sort by creation order, otherwise by name.

##### Iterate through the non-attribute objects

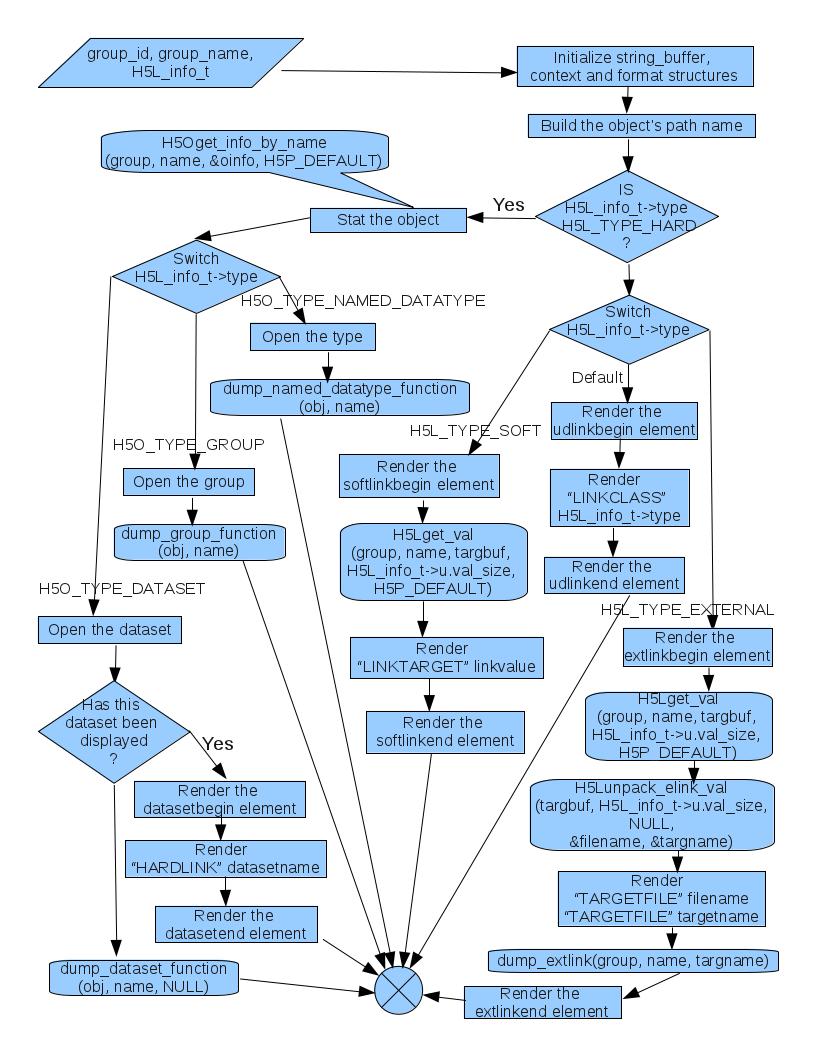
The function **dump\_all\_cb** is used as the callback in the H5Literate call. If there is a request to do H5\_INDEX\_CRT\_ORDER and tracking order is set in the group for attributes, then, sort by creation order, otherwise by name.

#### dump\_attr\_cb



This function displays the ATTRIBUTE information and the objects contained by the attribute. This function is a context wrapper for the h5tools\_dump\_attribute function that prints the attribute content. This function is used by the H5Aiterate calls in the functions; **dump\_group**, **dump\_dataset**, and **dump\_named\_datatype**. Also used by the XML **dump\_group** and **dump\_dataset** H5Aiterate calls.

#### dump\_all\_cb



This function displays the object information and the objects contained by this objects link information.

If the link is of type H5L\_TYPE\_HARD, the object info type (group, dataset, named\_datatype) determines which **dump\_function\_table** function is called; **dump\_group**, **dump\_dataset**, or **dump\_named\_datatype**. Else the value of the link type is printed; SOFTLINK is the LINKTARGET, EXTERNALLINK is the TARGETFILE and TARGETPATH, or the user defined link name and LINKCLASS.

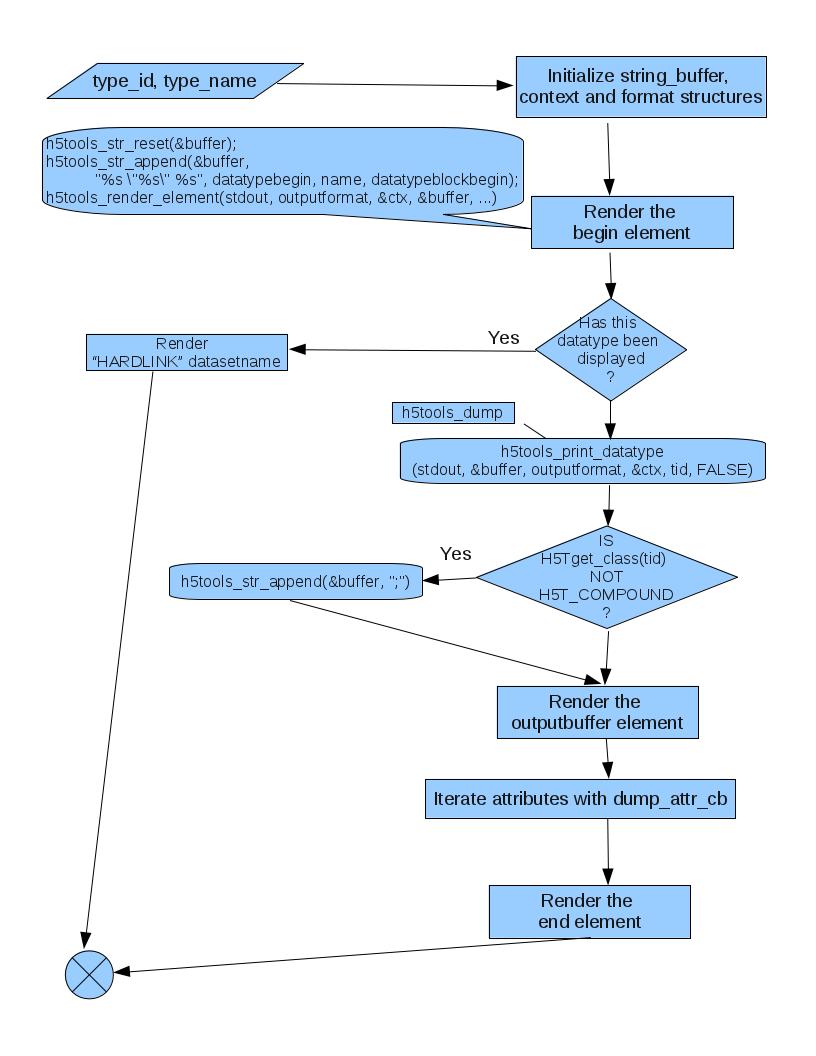
#### dump\_dataspace

This function is a context wrapper for the printing of dataspace information. Currently only the XML attribute/dataset dump procedure is using this function.

#### dump\_datatype

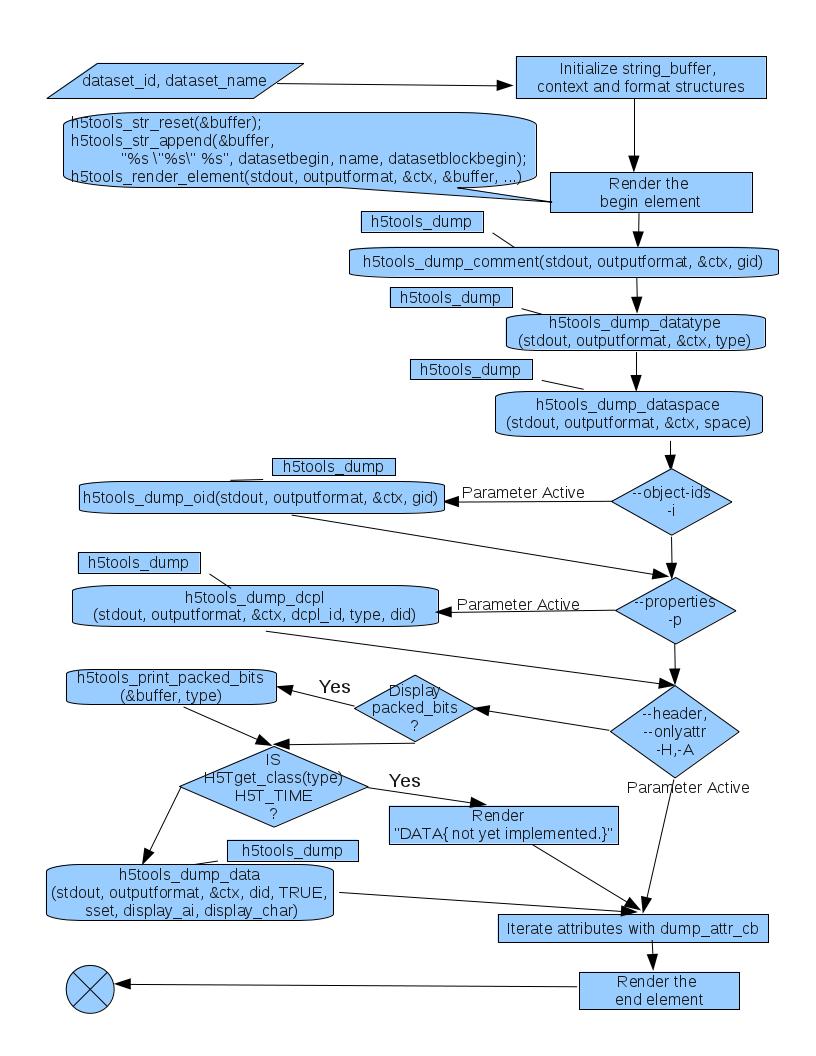
This function is a context wrapper for the printing of datatype information. Currently only the XML attribute/dataset dump procedure is using this function.

#### dump\_named\_datatype



This function displays the datatype information and iterates through the attributes contained by this object (using **dump\_attr\_cb** function). If there is a request to do H5\_INDEX\_CRT\_ORDER and tracking order is set in the group for attributes, then, sort by creation order, otherwise by name.

#### dump\_dataset

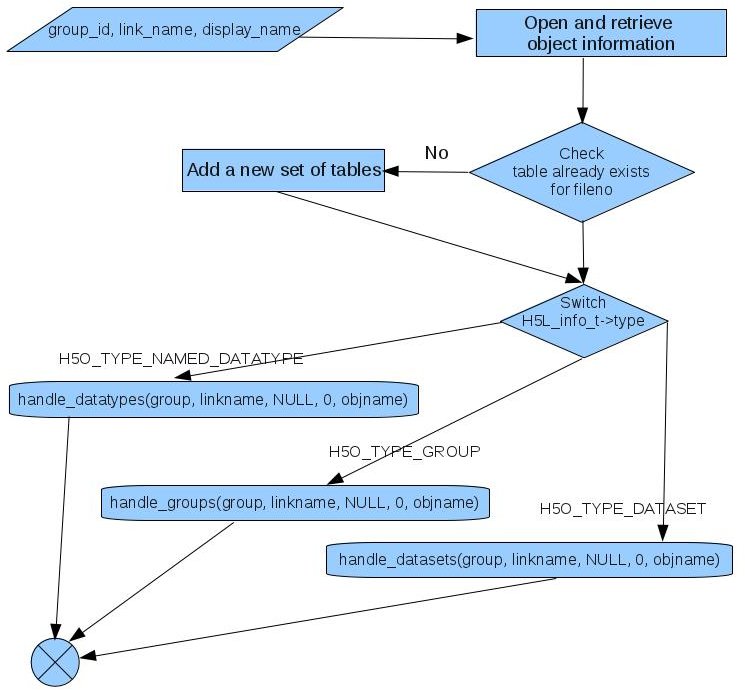


This function creates a context then displays the datatype, dataspace, optional dcpl, and optional data information using the h5tools library. Finally this function iterates through the attributes contained by this object (using **dump\_attr\_cb** function). If there is a request to do H5\_INDEX\_CRT\_ORDER and tracking order is set in the group for attributes, then, sort by creation order, otherwise by name.

#### dump\_data

This function is a context wrapper for the printing of data information. Currently only the XML attribute/dataset dump procedure is using this function.

### dump\_extlink



This function uses the handle\_xxxx functions to dump the selected group, dataset or datatype of the external link.

### handle individual objects

The handle\_xxxx functions are initialized by the command parser whenever the following parameters are encountered; -a(--attribute), -d(--dataset), -g(--group), -l(--soft-link), -t(--datatype). The structure the command line parser uses is:

/\* a structure for handling the order command-line parameters come in \*/

struct handler\_t {

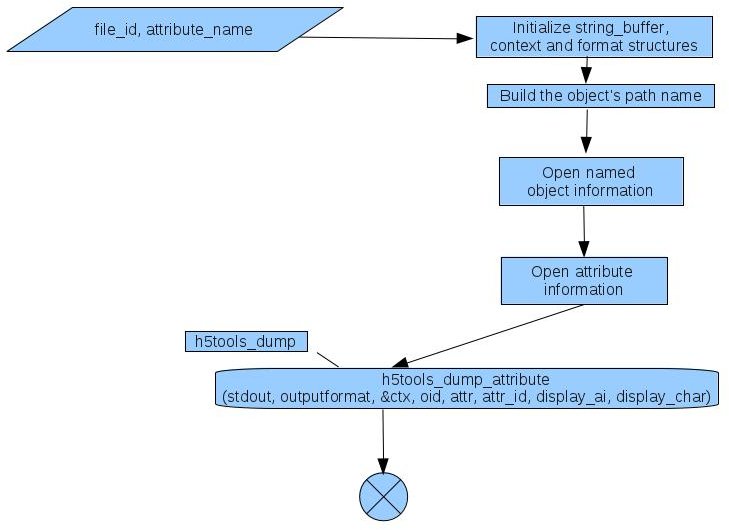
void (\*func)(hid\_t, const char \*, void \*, int, const char \*);

char \*obj;

struct subset\_t \*subset\_info;

};

#### handle\_attributes



This function initializes the context and format structures and opens the attribute before calling the h5tools library to output the attribute information.

#### handle\_datasets

#### handle\_datatypes

#### handle\_groups

#### handle\_links

# h5tools\_dump functions

# Reference

* RFC: Code Refactoring for h5dump at <https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/h5dump/h5dump_code_refactoring_v2.pdf>
* BNF: DDL in BNF for HDF5 at

<http://www.hdfgroup.org/HDF5/doc/ddl.html>

* XML: Document Type Definition (DTD) for HDF5 at

<http://www.hdfgroup.org/HDF5/XML/DTD/HDF5-File.dtd>

# Appendix

typedef struct h5tools\_context\_t {

size\_t cur\_column; /\*current column for output \*/

size\_t cur\_elmt; /\*current element/output line \*/

int need\_prefix; /\*is line prefix needed? \*/

unsigned ndims; /\*dimensionality \*/

hsize\_t p\_min\_idx[H5S\_MAX\_RANK]; /\*min selected index \*/

hsize\_t p\_max\_idx[H5S\_MAX\_RANK]; /\*max selected index \*/

int prev\_multiline; /\*was prev datum multiline? \*/

size\_t prev\_prefix\_len; /\*length of previous prefix \*/

int continuation; /\*continuation of previous data?\*/

hsize\_t size\_last\_dim; /\*the size of the last dimension,

\*needed so we can break after each

\*row \*/

int indent\_level; /\*the number of times we need some

\*extra indentation \*/

int default\_indent\_level; /\*this is used when the indent level gets changed \*/

hsize\_t acc[H5S\_MAX\_RANK]; /\* accumulator position \*/

hsize\_t pos[H5S\_MAX\_RANK]; /\* matrix position \*/

hsize\_t sm\_pos; /\* current stripmine element position \*/

} h5tools\_context\_t;

# Revision History

|  |  |
| --- | --- |
|  |  |
| *September 13, 2011:* | Version 1 new document split from RFC THG 2011-06-14.v1 |
| *September 19, 2011:* | Rewrote opening introduction |
| *October 6, 2011* | Version 2 added flow diagrams to dump\_function\_table descriptions |