HDF5 Metadata

Library and User Metadata

Version 3

Frank Baker

Background information for reviewers:

The contents of this file will be eventually appear in two separate publications:

* The first section (1 page) is a short, high-level description of HDF5 metadata. It is targeted for inclusion in a user guide chapter and was derived from text developed for the h5diff comparison specification document.
* The second section (5 pages) is a more comprehensive discussion. This is targeted for use as a stand-alone document in "Advanced Topics in HDF5."

These texts include a shift in terminology.

* Library metadata has previously been discussed as "structural metadata," but user metadata can also be structural; the key is that library metadata is automatically generated by the library.
* Static user metadata normally is set only once and can be changed only when rewriting an object or file; dynamic user metadata is designed to be changed at will. These two categories of user metadata have previously been discussed jointly as "application metadata," but it has become clear that the static versus dynamic distinction is worth discussing.

Feedback on this change of terminology is encouraged.

Quincey has observed that there’s also static and dynamic library metadata. For example, many object header messages are static, but the B-trees are dynamic. Quincey went on to say, “I don’t know if it’s worthwhile making that really clear though…” and I tend to agree. Users have such limited interaction with library metadata that this is not something we should devote much attention to, if any. If it is mentioned at all, a statement such as this should suffice:

Library metadata, like user metadata, has static and dynamic forms. For example, many object header entries are static, but the B-trees are dynamic. Users, however, have so little interaction with library metadata that this is not a useful distinction.

Your thoughts on any of these issues are welcome.

When published, this material will be in HTML and the embedded URLs converted to relative hyperlinks.

Thanks for your time. -- Frank

# HDF5 Metadata -- Overview

HDF5 files contain two types of metadata: library metadata and user metadata.

Library metadata is generated by the HDF5 Library to describe the structure of the file and structure and contents of objects in the file.  For example, library metadata includes information such as:

* A header block (the superblock) that sets up the file, sets up the initial structures, and identifies the file as a valid HDF5 file
* B-trees that describe the location of and provide access to groups and members of groups
* Object headers that set up objects in an HDF5 file

HDF5 natively interprets and understands library metadata.  Library metadata is always present; even an otherwise-empty file must contain certain metadata to be a valid HDF5 file.

User metadata is provided by, and in many cases defined by, a user application; is often stored in an HDF5 attribute; and may describe virtually anything.  Examples include:

* Minimum and maximum valid values in a dataset
* Conditions under which data was collected
* Data history and/or provenance
* Relationships among datasets
* Scales or other interpretive information

HDF5 does not natively understand most forms of user metadata; user metadata that the library does not natively understand must be interpreted by the application.  User metadata is optional but very commonly used.

# HDF5 Metadata – Detailed Review

HDF5 files can contain several types of metadata:

* Library metadata
* Static user metadata
* Dynamic user metadata

## Library Metadata

Library metadata is metadata that the user doesn't have any direct interaction with or control over. It is generated by the HDF5 Library to describe the structure of the file and structure and contents of objects in the file.  For example, library metadata includes information such as:

* Most elements of the header block (superblock), which sets up the file, sets up the initial structures, and identifies the file as a valid HDF5 file
* Object headers, which set up objects in an HDF5 file
* B-trees that describe the location of and provide access to groups and members of groups

HDF5 natively interprets and understands library metadata.  Library metadata is always present; even an otherwise-empty file must contain a superblock and a root group object header to be a valid HDF5 file.

## Dynamic and Static User Metadata

User metadata is defined and provided by the user application.

HDF5 does not always natively understand user metadata; much of it must be understood and interpreted by the application.  For example, the only thing the library understands in the dynamic user metadata list below is the extent of the dataset in the last bullet.

User metadata is technically optional but is almost universally used.

### Static user metadata

Static user metadata is information that the user can control but that is not generally dynamic.  It is stored in the file superblock or an object header and it does not usually change through the life of a file or object.  Examples of static user metadata include:

* Property lists: For example, H5Pset\_fapl\_family sets a file access property specifying that file I/O will use the family driver.
* Link names
* A dataset's datatype and dataspace (modulo the potential to extend or shrink it)
* Dataset fill values
* Dataset or group storage properties

Static user metadata does not usually change through the life of a file or object.  In some cases, it just doesn't tend to change; for example, the name of a hard link to an object can be changed only by creating a new hard link and removing the old link. In the more general case, static user metadata can be changed only when making a new copy of an HDF5 file or object.  For example, file creation and dataset creation properties can be changed only when making a new copy of a file or dataset, respectively.

### Dynamic user metadata

Dynamic user metadata is metadata that the user or application can change at will.  It is often stored in an HDF5 attribute, may describe virtually anything, and can easily change over time.  For example:

* Minimum and maximum valid values in a dataset
* Conditions under which data was collected
* Data history and/or provenance
* Relationships among datasets
* Scales or other interpretive information
* The extent of a chunked dataset within the bounds of its maximum extent

## Metadata Types and Mechanisms

Table 1 lists several examples of each type of HDF5 metadata, where it is stored, how it is set, and whether it is natively interpreted by the HDF5 library or must be interpreted by the user application. This is a representative subset, not a complete list of HDF5 metadata.

Table 1. Examples of HDF5 metadata by type

| Element | Where stored | How set | Interpreted by |
| --- | --- | --- | --- |
| Library metadata |  |  |  |
| Superblock | Header block at beginning of file | Created with file; always present | HDF5 library |
| File driver information | Superblock and driver information block | H5Pset\_fapl\_\* | HDF5 library |
| B-trees | At various locations within file | Library | HDF5 library |
| Object offsets | B-tree | Library | HDF5 library |
| Object headers | Header block for each object in an HDF5 file | Created with object; present as long as object exists | HDF5 library |
|  |  |  |  |
| Static user metadata |  |  |  |
| Dataset storage layout | Dataset object header | H5Pset\_layout | HDF5 library |
| Shared object header messages | Superblock and global heap | H5Pset\_shared\_mesg\_\* | HDF5 library |
| Link names and hierarchical structure | Group symbol table entries | H5G, H5L interfaces | Library |
| Permanent property lists | Dataset object header, data layout message, | H5P interface | Library |
| Transient property lists | Not stored | H5P interface | Library |
| Checksum | Dataset object header plus a checksum value accompanying each compressed dataset chunk | H5Pset\_fletcher32 | Library and application |
| Datatype | Dataset object header | H5T interface | HDF5 library |
| Dataspace  (contiguous dimensions or chunked maximum dimensions) | Dataset object header | H5S interface | HDF5 library |
|  |  |  |  |
| Dynamic user metadata |  |  |  |
| Min/max dataset values | Attribute(s) | H5A interface | Application |
| Data collection conditions | Attribute(s) | H5A interface | Application |
| Data provenance | Attribute(s) | H5A interface | Application |
| Object relationships  (other than hierarchical structure) | Attributes | H5A interface | Application |
| Measurement scales | Attribute(s) | H5A interface | Application |
| Dataspace  (current chunked dimensions) | Dataset object header | H5Dset\_extent | Library and application |

## Additional information

See the following texts for additional details, usage information, and examples.

### Properties

HDF5 property lists are used for both static and dynamic user metadata. Object creation property lists are static; they are stored with the object and cannot be changed without rewriting that object. Object access property lists are dynamic, or transient. They must be defined when an object is first created and redefined every time the object is opened; they are not stored.

Both static and dynamic user properties associated with the following classes of objects are discussed in specific chapters of The HDF5 User’s Guide (http://www.hdfgroup.org/HDF5/doc/UG/index.html):

Object Chapter  
File properties “The HDF5 File”Group properties “HDF5 Groups”   
Dataset properties “HDF5 Datasets”

The “H5P: Property List Interface” section (http://www.hdfgroup.org/HDF5/doc/RM/RM\_H5P.html) in the *HDF5 Reference Manual* lists and describes the usage details for all of the interfaces used to manage the above types of properties and several additional types.

### Attributes

HDF5 attributes, which offer nearly infinite flexibility for dynamic user metadata, are discussed in:

The “HDF5 Attributes” chapter of the *HDF5 User’s Guide*  
http://www.hdfgroup.org/HDF5/doc/UG/UG\_frame13Attributes.html

The “H5A: Attribute Interface” section in the HDF5 Reference Manual   
http://www.hdfgroup.org/HDF5/doc/RM/RM\_H5A.html

### Library metadata

Aside from the hierarchical structure of a file, library metadata is generally opaque to the user.

If a file’s structure is unknown, it can be determined through functions described in the “H5L: Link Interface,” “H5O: Object Interface,” and “H5G: Group Interface” sections of the HDF5 Reference Manual (http://www.hdfgroup.org/HDF5/doc/RM/RM\_H5Front.html).

### Managing metadata

#### Metadata cache

Significant performance gains can be achieved in certain circumstances by directly managing metadata I/O. This occurs most frequently in a high-performance computing (HPC) environment or when working with large data and complex access patterns. Managing the metadata cache can be a complex undertaking and should not be undertaken without careful study.

These issues are discussed in detail in “Metadata Caching in HDF5” (http://www.hdfgroup.org/HDF5/doc/Advanced/MetadataCache/index.html), a document in the collection Advance Topics in HDF5 (http://www.hdfgroup.org/HDF5/doc/Advanced.html).[[1]](#footnote-1)

The HDF5 functions used to manage metadata caching are described in the “H5F: File Interface” and “H5P: Property List Interface” sections of the HDF5 Reference Manual (http://www.hdfgroup.org/HDF5/doc/RM/RM\_H5Front.html). Look for functions with names containing the string ‘mdc’.

#### Metadata journaling

Some HDF5 applications can run for a very long time, sometimes for several days or even weeks. In such cases, an unexpected failure can cause the loss of all computed results that have not yet been written to storage. HDF5 provides the ability to periodically flush raw data to storage to guard against complete loss, but preserving metadata in the case of such a failure has been more problematic. HDF5 will introduce metadata journaling in HDF5 Release 1.10 to address this issue, making it possible to reconstruct metadata in the event of such an event.

In release 1.10, metadata journaling functions will be described in the HDF5 Reference Manual and a detailed discussion of their use will be included in the collection Advanced Topics in HDF5.

# Revision History

|  |  |
| --- | --- |
| *December 8, 2010:* | Version 1, uncirculated draft. |
| January 31, 2011: | Version 2, completeness and technical review. |
| February 8, 2011 | Version 3, limited internal circulation. |

# Glossary, Terminology

|  |  |
| --- | --- |
| **library metadata** | Metadata automatically created by the HDF5 library without input from the user application |
| **user metadata** | Metadata that is under the control of the user application |
| **dynamic user metadata** | User metadata that can change at will |
| **static user metadata** | User metadata that cannot be changed without rewriting an HDF5 file or object |
|  |  |
| **Deprecated terms:** |  |
| **structural metadata** |  |
| **application metadata** |  |

# See Also

HDF5 File Format Specification, <http://www.hdfgroup.org/HDF5/doc/H5.format.html>

HDF5 User’s Guide, <http://www.hdfgroup.org/HDF5/doc/UG/index.html>

HDF5 Reference Manual, <http://www.hdfgroup.org/HDF5/doc/RM/RM_H5Front.html>

1. The URLs in this paragraph will be valid only after the HDF5 Release 1.8.6 distribution, expected in mid-February 2011. [↑](#footnote-ref-1)