HDF5 File and Object Comparison Specification

Peter Cao
Frank Baker

This specification describes a set of rules for comparing two HDF5 files or two HDF5 objects. The intended purpose of the specification is to provide guidelines and information for developing tools or APIs to compare HDF5 files or objects.

1 Introduction

To compare two objects, one must know the expected equivalence relations and the differences to be searched for. These differences and equivalence relations have not previously been clearly defined for HDF5 files or objects, causing confusion in the implementation and interpretation of applications such as h5diff.

An HDF5 file appears to the user as a directed graph with three higher-level objects that are exposed by the HDF5 APIs: groups, datasets, and committed datatypes. The simplicity of the HDF5 model provides great flexibility with regard to what can be put in a file. At the same time, it creates challenges in determining how to compare two files or two objects. This document provides a clear definition of how two HDF5 files or objects should be compared by explicitly defining the equivalence relation and the differences to be examined.

The specification described in this document is intended for HDF5 files or objects only. It does not apply to non-HDF5 files and objects, including the user block in an HDF5 file.

One appropriate use of this specification would be in the development of an H5Ocompare API and/or a future version of h5diff (or a replacement tool).

2 Descriptions of HDF5 files and objects

This section briefly describes high level HDF5 objects and their metadata. For details, see the *HDF5 File Format Specification*[1] and related documents.[2]

2.1 Primary objects

HDF5 files are organized in a hierarchical structure, with three primary structures: groups, datasets, and committed datatypes.

[1] http://www.hdfgroup.org/HDF5/doc/H5.format.html
[2] http://www.hdfgroup.org/HDF5/doc/index.html

- ***HDF5 group***: a grouping structure containing zero or more links to groups or datasets and supporting metadata
- ***HDF5 dataset***: a multidimensional array of data elements and supporting metadata
- ***HDF5 committed datatype***: a committed datatype and supporting metadata

An object in HDF5 is identified either by link name within the group containing it or by its absolute path name (the path from the root group to the group containing the object plus the object's link name in that group).

2.2 Metadata

HDF5 files generally 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 (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
- Datatype, current and maximum array dimensions, and other features of a dataset
- Dataset properties such as storage layout, fill value, allocation time, or the use of filters

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 defined and provided by a user application, is often stored in an HDF5 attribute, and may describe virtually anything. 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

HDF5 does not natively understand user metadata; it must be understood and interpreted by the application. User metadata is technically optional but commonly used.

In both cases, these are just a few examples illustrating the range of possibilities.

3 Definition of equivalence

In this section, we define the HDF5 equivalence relation. Two files or objects are different if they are not equivalent.

3.1 Mathematic equivalence

Mathematic equivalence is defined in Wikipedia as follows: "A given binary relation "~" on a set A is said to be an equivalence relation if and only if it is reflexive, symmetric and transitive." Thus, an equivalence relation has three basic properties:

- Reflexivity: "a" ~ "a"
- Symmetry: if "a" ~ "b" then "b" ~ "a"
- Transitivity: if "a" ~ "b" and "b" ~ "c" then "a" ~ "c"

3.2 HDF5 equivalence

A single definition of equivalence or differences of two HDF5 files or two HDF5 objects is not adequate for all situations. Two definitions of equivalence are proposed here: strictly equivalent and loosely equivalent.

3.2.1 Strictly equivalent

Two objects or files are strictly equivalent if their content (all values or members) and metadata are the same. The three properties of the equivalence relation should be applied under strictly equivalent.

Under strict equivalence, two files are equal if their file structures, objects (all groups and datasets), and metadata are the same. Two groups are equivalent if structures, objects (all groups and datasets), and metadata are the same. Two datasets are equalivalent if their data values and metadata are the same.

3.2.2 Loosely equivalent

Two objects or files are loosely equivalent if they are equivalent under certain pre-determined conditions or options. The three properties of the equivalence relation may not all be applied to determine that two objects of files are loosely equivalent.

Below are some examples of comparison options for loosely equivalent. Other comparison options can be specified by an application.

Option A): Exclude certain objects when comparing two files or two groups

Option B): Compare only common objects

Option C): Ignore attributes or other metadata when comparing two objects

Option D): Compare only attributes or other metadata

4 Comparing HDF5 objects

This section describes how two HDF5 files or objects should be compared. Objects are identified by their link names when they are compared. Options, such as ignoring certain metadata, can be applied to comparisons (loosely equivalent).

4.1 Files

A valid HDF5 file contains a root group and metadata such as file creation properties. Comparing two files means comparing the root groups and the metadata of the two files. The group comparison is discussed in the following section; file metadata to be compared includes:

- File creation properties such as version information.

Special cases of file comparison include:

A) Itself: there should be no difference if a file is compared to itself.

B) Identical files: there should be no difference if two identical files are compared. This case is the same as case (A) except that the two files are two separate physical files in the system.

C) Empty files: a file is empty if it contains only a root group.

- If two empty files are compared, the result varies according to the comparison options. If metadata is ignored in the comparison, there should be no difference between two empty files. Otherwise, the result is determined by the metadata of the two files.
- If an empty file is compared with a non-empty file, the result varies according the comparison options. Under strictly equivalent, an empty file and a non-empty file should be different. Under certain loosely equivalent criteria, such as comparing only common objects, there will be no difference.

4.2 Groups

When two groups are compared, the links under the two groups and objects contained in the groups and having the same link names will be compared.

Group characteristics to be compared include:

- Group creation properties, such as creation order, group layout, etc.
- Attributes attached to the group
- Links

For objects contained in groups, characteristics to be compared are listed below by object type.

4.3 Datasets

An HDF5 dataset is an object that contains raw data and includes metadata that describes the data elements, data layout, and all other information necessary to write and read the stored data. For details, see the *HDF5 User's Guide*.[3]

[3] http://www.hdfgroup.org/HDF5/doc/UG/

Comparing datasets means comparing both the data values and metadata, unless an option excluding a specific comparison is given.

Dataset characteristics to be compared include:

- Dataset creation properties, such as storage layout, chunking, compression, fill value, etc.
- Attributes attached to the dataset
- Datatype
- Dataspace
- Raw data values

4.4 Links

A link is owned by a group and points to an existing object or a non-existing object (symbolic links only). Each link has a name, type, and value.

Link characteristics to be compared include:

- Name: A link name may be any string of ASCII or UTF-8 characters not containing a slash or a dot. The link name must be unique within each group.
- Type: A type specifies the link class. Valid types are hard link, soft link, and external link.
- Value: For soft links, this is the path to which the link points; for external and user-defined links, it is the link buffer.
- Creation properties, such as link order.

4.5 Attributes

Attributes are metadata for the current object. An attribute is a small dataset; it has a name, a datatype, a dataspace, and raw data. Attributes are compared by name and comparing two attributes is similar to comparing two datasets.

Attribute characteristics to be compared include:

- Datatype
- Dataspace
- Value(s)

4.6 Datatypes

A datatype defines the datatype for each element of a dataset or a committed datatype for sharing between multiple datasets. A datatype can describe an atomic type like a fixed- or floating-point type or more complex types like a C struct (compound datatype), array (array datatype) or C++ vector (variable-length datatype). A datatype is defined by its class and class-specific properties.

Datatype characteristics to be compared include:

- Datatype class, e.g. Integer, Float, String, etc.

- Class-specific properties, e.g. size, signed or unsigned, byte order, etc.

4.7 Dataspaces

A dataspace describes the rank, that is, the number of dimensions, and the size of each dimension in the data object array.

Dataspace characteristics to be compared include:

- Rank
- Current dimension sizes
- Max dimension sizes

4.8 Raw data values

When two datasets or attributes are compared, their values will be examined and compared. Comparing data values can be straightforward or complicated depending on their datatype. Special values and data structures must be handled case by case. Below are some examples:

- Floating point values: To determine whether two floating point values, float1 and float2, are different, one cannot use the simple comparison of (float1 == float2). Two floating point values can be the same while (float1 == float2) may appear to differ because of floating point precision. In HDF5, a precision limit can be set when comparing floating point values; for details, see the "Default EPSILON values for comparing floating point data" RFC.[4]
- Not-a-Number (NaN): Two NaNs are always equal. A NaN and a regular number are always different.
- Infinity: Infinity should be treated as a regular number. Two infinity numbers with the same sign (+/-) are always equal. Two infinity numbers with different signs (+/-) are always different. An infinity number and a regular number are always different.
- Special datatypes: The comparison values of special datatypes, such as opaque and variable-length datatypes, can be complicated. Such coomparisons must be handled on a case-by-case basis.

Revision History

October 8, 2010:	Version 1 circulated for comment within The HDF Group.
November 4, 2010	Version 2 revised after h5diff specification meeting with HDF Group staff (participants: Peter Cao, Quincey Koziol, Elena Pourmal, Jonathan Kim, and Neil Fortner).

[4] https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/h5diff/RFC_h5diff_default_epsilon.pdf

Appendix: Background Material

5 Major issues of h5diff

h5diff is a command line tool that compares two HDF5 files or objects and reports the differences. The tool provides options on what to compare and how to compare. For details, see the online h5diff description.[5]

This section briefly describes some major issues regarding h5diff. It is not meant to be comprehensive.

5.1 Performance

A performance problem has been a known issue for h5diff and is under investigation. One performance issue is in the handling of NaN datatypes. The current version of h5diff checks NaN values by default and the operation of checking and comparing special values is very time consuming. For datasets without special values, the time was totally wasted. A "-N" option is provided so that users can skip checking for NaN values.

Another performance issue in h5diff arises in retrieving and checking datatype information for each data point. This can be a major problem for datasets with large number of data points since checking datatype information, such as H5Tequal() and H5Tget_member_type(), is very expensive.

5.2 File structure

The current version of h5diff does not provide options for strictly equivalent and loosely equivalent. h5diff compares objects ONLY if it finds common objects in both files. This h5diff behavior will also hide potential errors in when testing the h5copy or h5repack tools; for example, if h5copy generates an empty HDF5 file by mistake, h5diff, as is, will report that the erroroneous empty file is not different from the original file. Therefore, h5diff should report that an empty HDF5 file is different from a non-empty HDF5 file.

A proposal has been made to add a new option "-c" as "Contents mode. Objects in both files must match." In view of the common practice of other comparison tools (e.g., the cmp and diff tools on Unix systems), it is better to fix h5diff as described in the above paragraph rather than introduce a new flag.

5.3 Non-Comparable datasets

For a variety of reasons, h5diff does not compare some dataset objects. When this happens, h5diff prints "Some objects are not comparable" at the end of the program execution. In verbose mode, h5diff also prints the reason(s) why it did not perform the comparison. The following are examples of non-comparable datasets listed in the RFC on non-comparable objects:

- Empty datasets
- Different datatype classes or H5T_TIME class or H5T_COMPOUND class

[5] http://www.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Diff

- Different dataspace ranks
- Different dataspace dimensions
- Different order properties
- Different sign properties
- Invalid numeric operation in relative error calculation

For details, see the "Reporting of Non-Comparable Datasets by h5diff" RFC.[6]

[6] https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/h5diff/RFC_h5diff_NonComparable.pdf