Better Testing for HDF5 Tools

Peter Cao

This document describes issues regarding the testing of HDF5 tools, and proposes some steps to address urgent needs and provide long term solutions.

1 Introduction

A customer recently reported issues regarding problems in HDF5 command line tools: “I think it would be good to have someone on your team dream up some regression tests (these seem to be basic functionality) for all h5* tools (including the h5check tool you deliver as a separate tool), run them, and find the issues before we do.”.

Insufficient testing of HDF5 tools has been a known issue for some time. Although it has been identified as a recent focus area, not much progress has been made for a variety of reasons. As The HDF Group is committed to the highest quality for our products and services, we need to take immediate action to address this critical issue.

2 Importance of good testing

Testing has always been an important part of the tools development. Good tests help us in many ways, such as:

- Ensure that changes (new features, bug fixes, or new tools) deliver the desired improvement or new functionality.
- Make certain that changes do not break any existing functionality or introduce new bugs.
- Provide a good way to test the library APIs, beyond what is covered by the library test suite. Since the tool testing is features oriented, it provides a good way of black box testing for the library.
- Confirm that the tools correctly handle all of the objects that may be in an HDF5 file for the latest version of the format. Or, at least those that are not explicitly documented as being not supported.

3 Current issues/problems

Tools testing can be more complicated than the testing for individual APIs in the library, since tools involve more combination of actions and files. Some problems with the current method of testing tools are listed here, together with statements of what is desired:

- Insufficient testing – Sometimes new features or bug fixes have to be rushed out to meet urgent needs, with changes only partially tested. *Code should not be released until it has been thoroughly tested for all basic cases, and well-tested for more unusual cases.*
- Missing general guidelines/standards – We do not have general guidelines on how our software should be tested at difference levels (unit testing, integration testing, system testing, regression testing, acceptance testing, etc), and testing is often not included in the design of the tool work. *Testing cases should be planned ahead, according to standard guidelines and intended changes.*
- No general framework – We do not have a general framework for testing tools. Each tool or feature is tested in a different way, making test generation and execution very time consuming and inconsistent. Some tests are too complicated. Adding a test case sometimes takes more time than the changes in the code, not because we try to cover more test cases but because the current tests are not effective and efficient. *A framework for testing tools should be adopted.*
- Lack of comprehensive test files – Most of our current test files are limited to very basic cases, and do not cover the full range of features defined by the HDF5 format. This means many things go untested, and considerable time is spent generating specialized test files to exercise a particular feature. *A set of comprehensive test files that can be used both by the library and the tools should be developed.*
- In general, current practice is to develop tests for tools based on changes to the tools and not on changes to the HDF5 library or format. As a consequence, new library or format features are often not adequately handled by the tools. *There should be a tighter coordination between introduction of library and format features and tools, and tool tests need to be updated when new library and format features are added.*

4 Recommendations for improvement

Our goal is to address all of the issues identified in Section 3. Because of limited staff resources, we break the tasks into two stages: address the urgent needs and provide long-term solutions.

4.1 Address urgent needs

The customer quoted in Section 1 has requested good regression tests for external links in HDF5 tools. We will start to address this urgent need with the steps below:

- Investigate what tools and features were implemented for supporting external/soft links
- Analyze the performance for h5ls, h5repack, h5diff, and h5copy on datasets with various layouts
- Design test cases for features/tools
- Create a set of test files that will include all the planned test cases
- Improve performance on HDF5 tools
- Have a set of test files/cases that will replicate the customer's files/needs

- Add those tests to the current tools test suite, and fix bugs caught by the tests

4.2 Provide long-term solutions

Long term solutions will involve changes to the testing for both the library and the tools. Our goal is to bring the quality of the tools to the same level of the library. Testing will be a good start. The following is a list of suggestions for improvement:

- Refuse to release software that has not been thoroughly tested, allocating sufficient time and resources to allow for all phases of the development process: design, coding, documentation, and testing.
- Provide testing guidelines and enforce them at design stage, making sure important test cases are included in the design or RFC for tool updates or new tools.
- Build a general framework for testing.
    - The general framework can be written scripts or XML so that it can be used on different platforms.
    - The framework should be used both in the library development and in the tools development.
- Apply regression tests from the library to the tools development if applicable.
- Have a set of comprehensive test files and a set of customer/application oriented files.
- When changes to the HDF5 library or format are considered, include consideration of tool changes and tool tests in the planning, implementation, and documentation.
    - If the tools cannot be updated to fully support the new features, they must be updated to fail gracefully, and the tool documentation must be updated to state that the feature is not supported.
    - For tools that will be updated, sufficient resources must be allocated to design, implement, document, and fully test the tools with the new features.

Revision History

September 30, 2010:	Version 1 circulated for comments with Ruth, Elena**,** and Quincey.
Octomber 7, 2010:	Version 2 revised after meeting with Elena and Quincey.