1. Introduction

This document outlines the HDF Group’s policy on accepting contributions of bug fixes and new features for inclusion in HDF5 library releases. In essence, this policy is simply a formalization of the HDF Group’s current policies and procedures for incorporating internally generated code – and as such, it should be expected to evolve with these internal policies and procedures.

While this policy should be applied with flexibility as dictated by circumstances, there should be well considered reasons for any deviations.

At present, several types of code contributions are contemplated:

1) Localized bug fixes.
2) Bug fixes that require architectural changes to the HDF5 Library.
3) New Features.

Policies for these different classes of contributions are discussed in separate sections below.

Note that no contribution may be accepted unless the donor has signed the Contributor Agreement (which may be obtained by emailing help@hdfgroup.org). The agreement need only be signed, and is valid for all future code contributions.

A contributor should request access to the HDF5 source code repository. Email TBD@hdfgroup.org and expected to follow the procedure described in this document.

2. Localized bugs

For the purposes of this document, localized bug fixes are bug fixes that repair coding errors only – and thus do not require changes to the architecture of the library.

Before such fixes are accepted, the following conditions should be met:

1) The bug is fully described in a Jira bug report, along with a protocol for exposing it, and a discussion of its solution.
2) The bug fix is accompanied with additions to the HDF5 library regression test suite that demonstrates the correctness of the bug fix.

3) The bug fix and its regression tests have been integrated into an up-to-date checkout of the develop HDF5 branch in Bitbucket (https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5/browse), which passes regression tests for basic HDF5 configurations (i.e., serial and parallel, debug and production modes on at least two different operating systems). <<need to spell out configure flags or point to public Confluence>>

4) A pull request has been made for the bug fix, with dedicated internal HDF5 developers listed as reviewers, and it has been approved by at least one of them.

5) Release.txt has been updated to include a description of the bug and its fix.


7) Daily test (including performance regression tests) after merge does not expose significant issues with the bug fix.

8) << We need to discuss merging of bug fixes into maintenance branches>>>

3. **Bug Fixes that Require Architectural Changes to the HDF5 Library**

Bug fixes that result in architectural changes to the HDF5 library require more careful handling. Before such fixes are accepted, the following conditions should be met:

1) The bug is fully described in a Jira bug report, along with a protocol for expressing it.

2) An RFC exists, which describes the current state of the relevant sections of the HDF5 library along with the proposed architectural changes needed to effect the bug fix.

3) The RFC has been discussed in one or more code review sessions, and all substantive questions and/or objections have been dealt with.

4) The code implementing the bug is fix accompanied with additions to the HDF5 library regression test suite that demonstrate the correctness of the bug fix.
5) The bug fix and its regression tests have been integrated into an up to date branch off the develop HDF5 branch, and passes the full set of regression tests (including performance tests) for the HDF5 library.

6) The code implementing the bug fix has been reviewed by at least two HDF5 library developers, and all substantive issues have been addressed.

7) A pull request has been made for the bug fix, with several HDF5 developers listed as reviewers, and it has been approved by at least one of them.

8) Release.txt has been updated to include a description of the bug and its fix.


4. New Features

Procedurally, there is little difference between our management of bug fixes requiring architectural changes to the HDF5 library, and our management of new features.

Before donations of new features are accepted for inclusion in HDF5 library releases, the following conditions should be met

1) An RFC exists, which describes the new feature. At a minimum, this RFC must address the following points:

   a. What is the motivation for the feature?

   b. What are the functional requirements of the new feature?

   c. If appropriate, what are the performance requirements for the new feature?

   d. How does it address these functional (and possibly performance) requirements?

   e. What additions / changes to the API are required?

   f. What if any file format changes are required?

   g. What is the current state of the relevant sections of the library, and how does the implementation of the new feature modify them?
The document should include detailed discussions of any algorithms and/or data structures used by the new feature (either directly, or by reference to readily accessible sources). The rule of thumb is that this discussion must address all elements of the implementation that are not readily grasped via inspection of the code.

h. How is the code implementing the new feature organized?
   In particular, indicate where the various functional elements of the new feature are implemented in the code base.

i. How is the new feature tested (including performance testing if appropriate)?

j. How is the test code organized?

k. How the new feature will affect the tools functionality? Any changes are planned for the tools?

l. Will the new feature affect backward/forward File Format and API compatibility?

2) The RFC has been discussed in one or more code review sessions, and all substantive questions and/or objections have been dealt with.

3) The code implementing the new feature and its associated test code exists on a feature branch that is up to date with the develop branch. Where appropriate, there should also be performance regression tests for the new feature.

4) The feature branch passes the full set of regression tests for the HDF5 library.

5) The code implementing the new feature has been reviewed by at least two HDF5 library developers, and all substantive issues have been addressed.

6) A pull request has been made for the new feature, with several internal HDF5 developers listed as reviewers, and it has been approved by at least one of them.

7) Release.txt has been updated to include a description of the new feature.


9) Blog post describing new feature is created.
For new features that are developed internally, there are a number of questions that are typically addressed before work starts on the feature. In the case of contributed features, these questions must still be addressed, albeit frequently after the fact.

1) On balance, is the feature of benefit to the HDF5 user community?

2) Does the feature as implemented have a positive or negative effect on the performance of the HDF5 library?

3) Does the feature as implemented have a positive or negative effect on the maintainability of the HDF5 library?

4) Is the feature as implemented as modular as is reasonably possible, given performance requirements?

5) How difficult is it to maintain the feature? Can we afford to do so?

6) Does the new feature facilitate or hinder future development of the HDF5 Library?

7) Is there an alternate way of satisfying the functional requirements that is markedly superior by any or all of the above metrics to that used by the donated feature?

What constitutes acceptable answers to the above questions is a judgment call. However, the point is that we must examine donated new features with as much care as we examine proposed new features for internal development, and accept them only if we are convinced that their value exceeds their cost.