RFC: How to deprecate and remove a feature in HDF

The HDF Group

This document summarizes the process of deprecating and removing a feature in HDF software.

A feature can be a function or a set of functions or subroutines in the HDF libraries, a command flag of command-line utility or the utility itself, a GUI feature in HDFView, a configure or CMake option, etc., i.e., anything that affects behavior or software or how it is used.

# Introduction

The HDF software has been under development and maintenance for the past 25 years. During this time we fixed numerous bugs, improved architecture and redesigned software internals, made enhancements to the HDF libraries to address performance and to improve maintainability, and added many new features. At the same time, several features became obsolete, were replaced with the new functionality, or just removed.

Every time we do changes to our software it becomes a disruption to our users and ourselves.

To minimize the disruptive effect of feature deprecation and further removal we should agree on basic principles and have a procedure to follow. This document addresses both. It also documents current deprecation practices for the HDF software packages and components.

# Feature deprecation

We say that a feature *is deprecated* in HDF5 if it was superseded by another feature and/or is not recommended for usage any more. Deprecated feature can be removed from the future versions of software.

## Why do we deprecate a feature?

Here are several reasons why a feature may be deprecated. The list is not exhaustive.

1. New more powerful, better-engineered, more general feature was introduced.
2. There was an error in design or implementation “beyond repair”.
3. It was a wrong thing to have it in the first place.
4. No one uses the feature anymore for different reasons (e.g., hardware for which it was created is gone).
5. New standard such as new naming convention was implemented.
6. The feature is not maintainable.
7. The feature encourages bad programming.
8. The feature is considered for removal in the future releases of software.

## When do we deprecate a feature?

When one of the conditions described in 2.1 is met, a feature may be recommended for deprecation. Deprecation may occur in any type of the releases including maintenance release. For description of the HDF5 library versioning see [1].

Planning for a release, especially for a minor or major release is a good time to evaluate functions for deprecation and initiate a deprecation process. Deprecation in a maintenance release should be considered as an exception and treated with a special care.

## What are depreciation principles?

We should follow several simple principles when we deprecate a feature.

* Source
  + A feature should easily identified and found in the source.
  + A feature can be “configured out” for compilation.
  + A feature can be easily removed from the source.
* Documentation
  + Deprecation request and following up work should be documented in JIRA.
  + A feature has to be clearly documented as deprecated in all user and internal documentation including RELEASE.txt file.
  + If applicable, documentation how to migrate to a new feature is provided.
* Outreach
  + It is required to identify major stakeholders and get their input on deprecation.
  + Deprecation should be announced at least six months before the action takes place.

## Who makes a recommendation and final decision?

Anyone can make a recommendation for feature deprecation, i.e., developers, maintainers and users. Recommendation should always include the reason why the feature should be deprecated. The recommendation should be entered into JIRA and assigned to a person who is responsible for the software maintenance.

The assigned person reviews the recommendation and proposes deprecation path according to the deprecation policy for the specific software or software component. The recommendation should be presented as an RFC and should be circulated within The HDF Group, identified stakeholders and community for the comments and final recommendations.

After recommendations are finalized, deprecation time line is announced and technical tasks for deprecation are scheduled.

All activities on feature deprecation should be documented in JIRA. All documents created during the process should be attached to JIRA.

# The deprecation process

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Who?** | **What?** | **When?** |
| 1 | Anyone | Proposes deprecation via sending email to [help@hdfgroup.org](mailto:help@hdfgroup.org) or directly entering request into JIRA | Anytime.  Planning time for the releases. |
| 2 | Person responsible for JIRA | Assigns a ticket to function maintainer | When tickets are assigned (on a weekly basis). |
| 3 | Function maintainer | Identifies relevant stakeholders  Creates RFC with deprecation proposal  Circulates for comments  Reworks the proposal based on the feedback  Creates final proposal | As scheduled. |
| 4 | Function maintainer | Schedules final review and gets and documents consensus from The HDF Group developers, and stakeholders | As scheduled. |
| 5 | Function maintainer | Assures that deprecation planning announcement goes out | At least six month when deprecation takes place. |
| 6 | Function maintainer | Follows software specific procedure to deprecate the function | As schedule for the release in which the function is deprecated. |
| 7 | Documentation team | Makes sure the function is documented appropriately | As schedule for the release in which the function is deprecated. |
| 8 | Outreach team | Announces function deprecation | At the time of the release in which deprecation happened. |
|  |  |  |  |

# Depreciation by software package

This section describes specific deprecation approach that is used now for each software package and its components.

## HDF5

### C library

### C++ library

### Fortran Library

### Command-line utilities

### High-Level C, Fortran and C++ Fortran Libraries

## HDF4

## HDF-JAVA

# References

1. “HDF5 Library Version Numbers” <http://www.hdfgroup.org/HDF5/doc/TechNotes/Version.html>

# Revision History

|  |  |
| --- | --- |
| *June 21, 2013:* | Version 1. Sent to Mike M. and Albert |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# References

1. “HDF5 Library Version Numbers” <http://www.hdfgroup.org/HDF5/doc/TechNotes/Version.html>