HDF5 Fortran Wrappers Maintenance:

Dropping Support for Non-Fortran 2003 Standard Compliant Compilers

M. Scot Breitenfeld and Elena Pourmal

The HDF5 Fortran library is a thin layer of Fortran and C wrapper functions on top of the HDF5 C library. When the HDF5 Fortran library was first released in 2001, the Fortran 90 compilers didn't provide a standard for handling interoperability with the C language. Therefore, special code had to be maintained to make the HDF5 Fortran library portable between UNIX and Windows platforms.

The introduction of the new Fortran 2003 standard enabled The HDF Group developers to extend Fortran support for advanced HDF5 library's features requiring, for example, usage of callback functions, iterations, and complex HDF5 datatypes. Also, Fortran 2003 introduced a standard for C interoperability.

Most –if not all– of the Fortran compilers available today are Fortran 2003 compliant. Hence, the HDF Group developers have taken advantage of the Fortran 2003 standard in order to greatly simplify the HDF5 Fortran library code. The changes described in this document became available in the HDF5 library version 1.10.0 released in March 2016.

As a result of these changes, the HDF5 1.10 Fortran library **requires a Fortran compiler that has a specified subset of the Fortran 2003 features implemented.** However, the described changes are transparent to the existing HDF5 Fortran applications.

This document outlines the reasoning for and implications of the implemented change on the HDF5 Fortran library source code and provides guidance for future maintenance.

Introduction

The Fortran 2003 (F2003) standard greatly simplifies the interoperability with C and it: (1) improves the portability of the Fortran wrappers with the main C HDF5 library, (2) reduces the time invested in Fortran development and (3) reduces the effort in maintaining the Fortran wrappers. The HDF Group made use of these improvements in the v1.10.0 release of HDF5. The majority, if not all, of the current compilers now have, and have had for some time, the C interoperability features implemented. Therefore, **The HDF Group requires a minimum implementation of F2003 interoperability features in order to compile HDF5 v1.10.0 and later.**

Fortran practices in HDF5 1.8 and earlier

Currently, HDF5 v1.8 handles non-F2003 compilation using configure option –*enable-fortran*. The

F2003 features are enabled if both the configure options *–enable-fortran* and *–enable-fortan2003* are used. The Fortran files are organized in the *fortran/src* directory as follows:

- Source files ending in *.f90* contain APIs which are both F90 and F2003 compatible,
- Source files ending in *_F90.f90* contain F90 APIs which also have a F2003 equivalent (if available) API,
- Source files ending in *_F03.f90* contain F2003 APIs that require a F2003 compiler.

The appropriate set of files to compile are chosen at build time depending on which set of configure flags were set. If, during configure of a F2003 build, the build process detects that the compiler does not meet the F2003 standard requirements then the build process is terminated with an error. The user then has to specify only *–enable-fortran* and rebuild.

Continued support practices in HDF5

It is important to note that **HDF5 continues support for F90/F95 standard compliant codes calling HDF5**. Except for the currently required use of a Fortran 90 module (i.e. USE HDF5), all F90/F95 standard compliant code will remain compatible with the HDF5 library. However, some newer HDF5 APIs will require F2003 standard features in order to be used. All Fortran codes using HDF5 will not have to change any API currently supported. The adoption of F2003 in HDF5 should be transparent in terms of the impact on user's HDF5 Fortran codes.

Moving to the F2003 standard in HDF5 1.10

Required F2003 features

The Fortran compiler needs not to have the entire F2003 standard implemented in order to compile HDF5 v1.10. The list of required features needing to be implemented are:

- The ISO_C_BINDING module must be available,
- C_PTR, C_FUNPTR must be implemented,
- BIND(C) must be implemented.

Known compilers not meeting F2003 requirements

Known compilers not meeting the F2003 requirements as described in 2.1 are:

- gfortran version 4.1 and earlier.

Note, the HDF Group dropped support for those compilers as of May 15, 2015 after the HDF5 1.8.15 release.

Build behavior in v1.10

Specifying the compiler option *–enable-fortran* will compile all the Fortran wrappers. The option *–enable-fortran2003* will be **discontinued.**

Benefits of adopting F2003 standards

The maintainability of the Fortran wrappers improves: (1) the development of new wrappers and (2) reduces the complexity of the current implementation.

Use of BIND(C)

In HDF5 1.8 the Fortran wrappers naming convention with the C library are handled internally by configure and an internal HDF5 convention. For example, the Fortran API interface for h5awrite_f_c is:

```
INTERFACE
      INTEGER FUNCTION h5awrite_f_c(attr_id, mem_type_id, buf)
        USE, INTRINSIC :: ISO_C_BINDING, ONLY : c_ptr
        !DEC$IF DEFINED(HDF5F90_WINDOWS)
        !DEC$ATTRIBUTES C,reference,decorate,alias:'H5AWRITE_F_C'::h5awrite_f_c
        !DEC$ENDIF
          ...
      END FUNCTION h5awrite_f_c
END INTERFACE
```

The !DEC$ is required to provided portability between UNIX and Windows platforms with Intel Fortran compile. With the use of BIND(C) in HDF5 1.10, the new interface does not require the !DEC$ declarations.

```
INTERFACE
      INTEGER FUNCTION h5awrite_f_c(attr_id, mem_type_id, buf) BIND(C, NAME='h5awrite_f_c')
        USE, INTRINSIC :: ISO_C_BINDING, ONLY : c_ptr
          ...
      END FUNCTION h5awrite_f_c
END INTERFACE
```

Besides now being standard compliant, the name of the C API is also specified directly. Therefore, the current convention of defining a name space alias in H5f90proto.h,

```
#define nh5awrite_f_c       H5_FC_FUNC_(h5awrite_f_c, H5AWRITE_F_C)
H5_FCDLL int_f nh5awrite_f_c (hid_t_f *attr_id, hid_t_f *mem_type_id, void *buf);
```

is simplified by removing the need to use #define and by using the name of the C API wrappers directly,

```
H5_FCDLL h5awrite_f_c (hid_t_f *attr_id, hid_t_f *mem_type_id, void *buf);
```

Furthermore, most – if not all – of the C wrappers can be eliminated by calling the HDF5 C APIs directly from the Fortran wrappers. Doing so also enables Fortran programs to call the HDF5 C APIs directly, avoiding the Fortran wrappers all together, if needed. Finally, as discussed in Section 1.1, the need to maintain and develop both F90 and F2003 compliant wrappers is eliminated.

Implemented Changes to the Fortran Structure in HDF5 1.10

Use of BIND(C)

Switching to using BIND(C) requires:

- All current Fortran APIs to use the BIND(C, name=) convention,
- All !DEC$ lines are eliminated,
- The use of #define is eliminated,
- The C API matchs the name given in the BIND(C, name=) specification.

Removal of Dual Functioning Fortran Files and Future Maintenance

All duplicate Fortran 90 APIs, which have an equivalent Fortran 2003, were eliminated. The _F03.f90 APIs were combined with the .f90 files and the _F90.f90 files were eliminated. All the autotools and CMake related files were updated to reflect these changes.

Configure automatically generates the Fortran 90 APIs in order to handle backward compatibility of F90 APIs. At configure, all valid integer and real KINDs are determined, in addition to the maximum decimal precision for reals and floats in Fortran and C, respectively. The valid KINDs for integers and reals found by configure are used in *H5_buildiface.F90* (fortran/src) via the preprocess include file *H5config_f.inc*. During make, *H5_buildiface.F90* generates all the valid F90 KIND interfaces for *h5awrite_f, h5aread_f, h5dwrite_f, h5dread_f, h5pset_fill_value_f, h5pget_fill_value_f, h5pset_f, h5pget_f, h5pregister_f and h5pinsert_f* to handle up to, and including, rank 7 arrays. No new Fortran APIs should be added to H5_buildiface.F90 since new Fortran APIs should not use F90 specifications, but should instead use F2003. The source file generated by *H5_buildiface.F90* is *H5_gen.F90* and is the Fortran module “H5_GEN”. This module is included in the HDF5 module.

The Fortran HL implementation mirrors the low-level Fortran APIs as described above. During make, *H5HL_buildiface.F90* (hl/fortran/src) generates all the valid F90 KIND interfaces, found in *H5config_f.inc*, i.e., it generates all the valid F90 KIND interfaces *h5ltread_dataset_f, h5ltread_dataset_int_f, h5ltmake_dataset_int_f, h5ltmake_dataset_float_f, h5ltmake_dataset_double_f, h5ltread_dataset_float_f, h5ltread_dataset_double_f, h5tbwrite_field_name_f, h5tbwrite_field_index_f, h5tbread_field_index_f and h5tbinsert_field_f*. This implementation fixes the generic procedure violation in HDF5 1.8 when an 8-byte REAL is used as the default. The source file generated by *H5HL_buildiface.F90* is *H5LTff_gen.F90* and contains the modules “H5LT” and “H5TB”.

The maximum precision value is used in *H5match_types.c* to match the Fortran real KIND to an interoperable C data type.

Fortran File Preprocessing

Fortran files were renamed from *.f90* to *.F90* in order to indicate that the files should be pre-processed. In HDF5 1.8 no Fortran source code allows for preprocessing directives. The preprocessor directives allow for the inclusion of Fortran standard dependent intrinsic functions and features in a

common source file. This is opposed to the HDF5 1.8 practice of including separate source files, chosen at build time, depending on the availability of, or lack thereof, the intrinsic function or feature.

Acknowledgements

The HDF Group internally funded this work.

Revision History

February 5, 2015:	Version 1 circulated within The HDF Group Fortran developers.
April 3, 2015:	Version 2 circulated within The HDF Group Fortran developers.
April 9, 2015	Version 3 sent to The HDF Group HDF5 developers.
April 10, 2015	Version 4 sent to the Forum.
March 28, 2015	Version 5 updated with final implementation details.
April 5, 2016	Version 6 updated to take into consideration HDF5 1.10.0 release

1)