RFC: 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.

More recently, the introduction of the new Fortran 2003 standard enabled The HDF Group developers to extend Fortran support for advanced HDF5 library features requiring, for example, usage of callback functions, iterations, and complex HDF5 datatypes. Additionally, Fortran 2003 also 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 would like to take advantage of the Fortran 2003 standard in order to greatly simplify the HDF5 Fortran library code. The changes proposed in this document are slated for the HDF5 1.10.0 release in March 2016.

As a result of these changes, building the HDF5 Fortran library **will require** a Fortran compiler that has a specified subset of the Fortran 2003 features implemented. However, the proposed changes will be transparent to existing HDF5 Fortran applications. Furthermore, application codes themselves will not be required to use a Fortran 2003 compliant compiler unless they are calling HDF5 Fortran APIs that incorporate Fortran 2003 features.

This document describes the reasoning for and implications of the proposed change on the HDF5 Fortran library source code.

1 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 plans to make use of these improvements in the v1.10 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 plans on requiring a minimum implementation of F2003 interoperability features in order to compile HDF5 v1.10.

1.1 Current Fortran practices in HDF5

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 must specify only *–enable-fortran* and rebuild.

1.2 Continued support practices in HDF5

It is important to note that HDF5 is not dropping 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.

2 Moving to the F2003 standard

2.1 Required F2003 features

The Fortran compiler does not need to have the entire F2003 standard implemented in order to compile HDF5 v1.10. The list of required F2003 features is:

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

2.2 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 will drop support for those compilers as of May 15, 2015 after the HDF5 1.8.15 release, so this should not be an issue for the future 1.10 release.

2.3 Proposed build behavior in v1.10

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

2.4 Benefits of adopting F2003 standards

Dropping support for non-F2003 compilers when building the HDF5 Fortran wrappers will improve: (1) the development of new wrappers and (2) reduce the complexity of the current implementation.

2.4.1 Reduced complexity by the use of BIND(C)

Currently the Fortran wrappers' naming convention with the C library is handled by the configure script 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$ declarations are required to provided portability between UNIX and Windows platforms with Intel Fortran compiler. With the use of BIND(C), 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[1], 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);
```

Finally, as discussed in Section 1.1, the need to maintain and develop both F90 and F2003 compliant wrappers is eliminated.

3 Proposed Implementation Changes to the Fortran Structure

3.1 Use of BIND(C)

Switching to using BIND(C) requires:

- All current Fortran APIs to use the BIND(C, name=) convention,

[1] To be standard compliant when calling a C function from Fortran an interface needs to use BIND(C). The Fortran standard doesn't guarantee interoperability with C code if BIND(C) is not used.

- All !DEC$ lines will be eliminated,
- The use of #define will be eliminated,
- The C API will match the name given in the BIND(C, name=) specification.

3.2 Alternatives to calling the HDF C wrappers

An alternative to calling the C wrappers would be to eliminate the C wrappers entirely by calling the HDF5 C APIs directly from the Fortran wrappers. This would mean rewriting all 300+ C wrappers in Fortran. There are a few issues with doing this:

1. It discards robust and tested C wrappers and starts from essentially scratch developing new Fortran wrappers.
2. All C structures must have an equivalent Fortran structure, and this may not always be possible. We currently do not need to recreate all the C structures in Fortran. Additionally, this creates maintenance issues because if the C structure gets changed then it will also have to be updated in Fortran. Currently we copy C structures into Fortran structures; consequently, they don't have to match.
3. The Fortran wrappers might end up being more complex then the C wrappers they are replacing. It does not necessarily simplify the code doing everything in Fortran. It remains unclear if this would actually simplify the Fortran library or just shift the complexity from the C wrappers to the Fortran wrappers.
4. In terms of building the HDF5 Fortran library, it would simplify linking to C wrappers and then having to link to the main HDF5 C APIs, but it's uncertain what problems, if any, this would solve when building the Fortran Library.

It might be best to balance the two options in 3.1 and 3.2 by calling the C HDF5 APIs directly when it is straight forward to do so, but leave the more complicated C APIs called from a C wrapper.

3.3 Removal of Dual Functioning Fortran Files

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

3.4 Fortran File Preprocessing

Fortran files will be renamed from *.f90* to *.F90* in order to indicate that the files should be pre-processed. Currently, no Fortran source code allows for preprocessing directives. Although this change is not specifically needed for Fortran 2003, allowing for preprocessor directives will allow for the inclusion of Fortran standard dependent intrinsic functions and features in a common source file. This is opposed to the current 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 13, 2015	Version 4 sent to the Forum.
April 15, 2015	Version 5 sent to The HDF Group HDF5 developers.