HDF 4.2.15

Release Information

Version	HDF 4.2.15
Release Date	2020-02-27
Download	Download
Release Notes	Release Notes
Newsletter	Announcement

Files

NOTE
Please be aware that on Mac 10.13 and later, the `--enable-hdf4-xdr` (autotools) / `HDF4_BUILD_XDR_LIB:BOOL=ON` (CMake) option must be specified.

File	Operating System	Compilers	Comments	MD5 or sha256 Checksum
hdf-4.2.15.tar	Source release		Source tar file	hdf-4.2.15.md5
hdf-4.2.15.tar.gz	Source release		Gzipped source tar file (see NOTE)	"
hdf-4.2.15.tar.bz2	Source release		Bzipped source tar file (see NOTE)	"
hdf-4.2.15.zip	Source release		Windows zip file	"
CMake-hdf-4.2.15.tar.gz	CMake source release		Source file to build with CMake on Unix (See NOTE) See: Build Instructions	"
CMake-hdf-4.2.15.zip	CMake source release		Source file to build with CMake on Windows See: Build Instructions	"
hdf-4.2.15-linux-centos7-x86_64-shared.tar.gz	Linux 3.10 CentOS 7 x86_64	gcc and java 1.8+		hdf-4.2.15-linux-centos7-x86_64-shared.tar.gz.sha256
hdf-4.2.15-osx1013_64-clang.tar.gz	Darwin 17.7	Apple clang 10,0.0 and gfortran 6.3.0		hdf-4.2.15-osx1013_64-clang.tar.gz.sha256
hdf-4.2.15-win10_64-vs15.zip	Windows 10 64-bit	CMake VS 2017 C++, IVF, openjdk 11		hdf-4.2.15-win10_64-vs15.zip.sha256
hdf-4.2.15-win10_64-vs14.zip	Windows 10 64-bit	CMake VS 2015 C++, IVF, openjdk 11		hdf-4.2.15-win10_64-vs14.zip.sha256
hdf-4.2.15-win10_64-vs15-Intel.zip	Windows 10 64-bit	CMake VS 2017 w/ Intel C, Fortran 2019, openjdk 11		hdf-4.2.15-win10_64-vs15-Intel.zip.sha256

hdf-4.2.15-win10_64-vs14-Intel.zip	Windows 10 64-bit	CMake VS 2015 w/ Intel C, Fortran 2018, openjdk 11		hdf-4.2.15-win10_64-vs14-Intel.zip.sha256
hdf-4.2.15-win7_64-vs14.zip	Windows 7 64-bit	CMake VS 2015 C++, IVF, openjdk 11		hdf-4.2.15-win7_64-vs14.zip.sha256

Release Notes

```
HDF version 4.2.15 released on 2020-02-12
====================================================

INTRODUCTION

This document describes the differences between this release and the
HDF 4.2.14 release.  It is written for people who are familiar with
previous releases of HDF and wish to migrate to this version of HDF.

Note that the HDF4 documentation will be updated at the time of
each final release and can be found on the HDF4 support page at:

    https://portal.hdfgroup.org/display/HDF4/HDF4

The official HDF4 releases can be obtained from:

    https://portal.hdfgroup.org/display/support/Download+HDF4

If you have any questions or comments, please send them to the HDF Help Desk:

     help@hdfgroup.org

CONTENTS

- New features and changes
  -- Configuration
- Support for new platforms and compilers
- Bugs fixed since HDF 4.2.14
  -- Configuration
  -- Library
  -- Utilities
- Documentation
- Platforms tested
- Known problems


New features and changes
========================
    Configuration:
    -------------
    - Updated configuration for both build systems

      Synchronized configuration variables between build systems to create
      the same variables in h4config.h. Replaced the defines used in #ifdef
      blocks with the appropriate H4_XXX defines. Updated the internal xdr
      code to handle the different size of integers across platforms.
      Also removed the restriction on building 64-bit platforms. Users on
```

macOS platforms (10.13 and newer) must build with the hdf4 xdr by adding the following options during configuration:

```
  autotools add "--enable-hdf4-xdr"
  CMake add "HDF4_BUILD_XDR_LIB:BOOL=ON"
```

See INSTALL file for details on configure options.

(ADB - 2020/02/21)

- Update CMake for VS2019 support

  CMake added support for VS2019 in version 3.15. Changes to the CMake generator setting required changes to scripts. Also updated version references in CMake files as necessary.

  (ADB - 2019/11/18, HDFFR-1581)

- Update CMake tests to use FIXTURES

  CMake test fixtures allow setup/cleanup tests and other dependency requirements as properties for tests. This is more flexible for modern CMake code.

  (ADB - 2019/07/23, HDFFV-10529)

- Windows PDB files are always installed

  There are build configuration or flag settings for Windows that may not generate PDB files. If those files are not generated then the install utility will fail because those PDB files are not found. An optional variable, DISABLE_PDB_FILES, was added to not install PDB files.

  (ADB - 2019/07/17, HDFFV-10424)

- Add mingw CMake support with a toolchain file

  There has been a number of mingw issues that has been linked under HDFFV-10845. It has been decided to implement the CMake cross-compiling technique of toolchain files. We will use a linux platform with the mingw compiler stack for testing. Only the C language is fully supported, and the error tests are skipped. The C++ language works for static but shared builds has a shared library issue with the mingw Standard Exception Handling library, which is not available on Windows. Fortran has a common cross-compile problem with the fortran configure tests.

  (ADB - 2019/07/12, HDFFV-10845, HDFFV-10595)

- Windows PDB files are installed incorrectly

  For static builds, the PDB files for windows should be installed next to the static libraries in the lib folder. Also the debug versions of libraries and PDB files are now correctly built using the default CMAKE_DEBUG_POSTFIX setting.

  (ADB - 2019/07/09, HDFFV-10581)

- Add option to build only shared libs

  A request was made to prevent building static libraries and only build shared. A new option was added to CMake, ONLY_SHARED_LIBS, which will

```
      skip building static libraries. Certain utility functions will build with
      static libs but are not published. Tests are adjusted to use the correct
      libraries depending on SHARED/STATIC settings.

      (ADB - 2019/06/12, HDFFV-10805)



Support for new platforms and compilers
=======================================

    - CMake added support for VS2019 in version 3.15. Updated scripts.

    - macOS 10.13.6, Darwin 17.7.0 with Apple clang LLVM version 10.0.0

    - macOS 10.14.6, Darwin 18.7.0 with Apple clang LLVM version 10.0.1


Bugs fixed since HDF 4.2.14
===========================

    - netCDF test failed on multiple Mac

      The netCDF test in the hdf4 library had failed either with segfault
      or incorrect reading data on multiple versions of Mac machines, when
      compiling with -O2.  The segfault appeared to happen during the use
      of various system XDR functions.  In previous releases, -O0 level
      optimization was used to avoid this problem.  In this release, the
      hdf4's internal xdr library has been updated to handle the different
      size of integers across platforms, based on the newer tirpc library
      code and netCDF xdr code.

      (ADB/BMR/LRK - 2020/02/15, EED-439)

    - Several memory leaks are fixed

      There were memory leaks caused by the allocated xdr structure not released
      and several missing to free allocated buffers in tests.  These are now fixed.

      (BMR - 2019/11/15)


Documentation
=============


Platforms tested
================
This version has been tested in the following platforms:

(Format:
   uname -s, uname -r
   uname -v, uname -p, uname -m)



   Linux 2.6.32-754.11.1.el6.x86_64  gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-16)
   #1 SMP, x86_64                    GNU Fortran (GCC) 4.4.7 20120313 (Red Hat
4.4.7-16)
```

```
   (mayll/platypus)                  icc (ICC) 17.0.0.098 Build 20160721
                                     ifort (IFORT) 17.0.0.098 Build 20160721
                                     pgcc and pgf90 17.10-0 64-bit target
                                         on x86-64 Linux -tp nehalem

   Linux, 3.10.0-327.18.2.el7.x86_64 GNU C (gcc) and Fortran (gfortran) compilers:
   #1 SMP x86_64, GNU/Linux             Version 4.8.5 20150623 (Red Hat 4.8.5-4)
   jelly/moohan)                        Version 4.9.3, 5.3.0, 6.2.0, 7.1.0, 8.3.0
                                     Intel(R) C (icc) and Fortran (ifort) compilers:

                                        Version 17.0.0.098 Build 20160721
                                     pgcc and pgf90 17.10-0 64-bit target
                                         on x86-64 Linux -tp haswell

   Linux, 2.6.32-573.18.1.el6.ppc64  GNU C (gcc) and Fortran (gfortran) compilers:
   #1 SMP, ppc64 (ostrich)             Version 4.4.7 20120313 (Red Hat 4.4.7-18)
                                     Fortran (xlf)  for Linux Version 15.1

   SunOS 5.11 (32- and 64-bit)       Sun C 5.12 SunOS_sparc 2011/11/16
   11.1, sparc, sun4v (emu)          Sun Fortran 95 8.6 SunOS_sparc 2011/11/16

    Windows 7 x64                    Visual Studio 2015 w/ Intel C, Fortran 2018
(cmake)

    Windows 10 x64                   Visual Studio 2015 w/ Intel C, Fortran 2018
(cmake)
                                     Visual Studio 2017 w/ Intel C, Fortran 2019

(cmake)
                                     Visual Studio 2019 w/ Intel C, Fortran 2019

(cmake)


   Mac OS X 10.11.5, Darwin, 15.6.0  Apple clang version 7.3 from Xcode 7.3
   x86_64                            gfortran GNU Fortran (GCC) 5.2.0
   (osx1011test)                     Intel icc and ifort version 15.0.3

   macOS Sierra 10.12.5, Darwin,     Apple clang version 8.1.0 from Xcode 8.3
   16.6.0, x86_64                    gfortran GNU Fortran (GCC) 7.4.0
   (kite)                            Intel icc and ifort version 17.0.2

   macOS 10.13.6, Darwin,            Apple clang LLVM version 10.0.0
   17.7.0, x86_64                    gfortran GNU Fortran (GCC) 6.3.0
   (bear)                            Intel icc and ifort version 19.0.4

   macOS 10.14.6, Darwin,            Apple clang LLVM version 10.0.1
   18.7.0, x86_64                    gfortran GNU Fortran (GCC) 6.3.0
   (bobcat)                          Intel icc version 19.0.4

   Fedora30 5.3.11-200.fc30.x86_64
   #1 SMP x86_64  GNU/Linux          gcc (GCC) 9.2.1 20190827 (Red Hat 9.2.1
20190827)
                                     GNU Fortran (GCC) 9.2.1 20190827 (Red Hat 9.2.1

20190827)
                                     (cmake and autotools)

   Linux, 4.15.0-1051-aws
   #53-Ubuntu SMP, x86_64, x86_64    gcc (Ubuntu 7.4.0-1ubuntu18.04.1) 7.4.0
                                     GNU Fortran (Ubuntu 7.4.0-1ubuntu18.04.1) 7.4.0
                                     (cmake and autotools)
```

```
Known problems
==============
o  Builds with the autotools option --enable-hdf4-xdr fail on Solaris and
   on IBM ppc64 with the xlc compiler.  The option should not be used on
   those platforms.

o  CMake files do not behave correctly with paths containing spaces.
   Do not use spaces in paths because the required escaping for handling spaces
   results in very complex and fragile build files.
   ADB - 2019/05/07

o  Several Fortran examples print "^@" when displaying strings (for example,
   names of the attributes). This happens because Fortran application
   doesn't know the length of the strings passed from the C library.
   EIP - 2015-01-11, HDFFR-1477

o  CMake fails to set the full path to the install location on Windows:
    The configuration file for examples, HDF4_Examples.cmake, must be updated
    with the correct value by editing the file or using the INSTALLDIR option.
    This issue is because of spaces in the path.
   ADB - 2014/02/03

o  CMake "make install" fails installing the tools:
    Use CPack to create an install package.
   ADB - 2014/02/03

o  CMake does not install these man pages:
    hdf.1, ncdump.1, ncgen.1
   AKC/BMR - 2014/02/02

o  For Mac OS X 10.7 Lion, 10.8 Mountain Lion, 10.9 Mavericks, 10.10 Yosemite,
   and 10.11 El Capitan, when compiling with -O2, some xdr functions might cause
   memory corruption.  This happened for GCC, Intel and Clang compilers.
   Currently, -O0 level optimization is used to avoid this problem.
   (HDFFR-1318,1327,1358,1425) EIP - 2013/02/05, BMR - 2016/06/24
   Update: This issue has been addressed in 4.2.15. BMR - 2020/02/24

o  On IBM PowerPC 64, hdftest fails when gcc 4.4.6 is used with -O3 optimization
   level.

o  When building in AIX systems, if CC is xlc with -qlanglvl=ansi, configure
   will fail when checking for the jpeglib.h header due to the duplicated
   macro definition of HAVE_STDLIB_H.  This is because some newer builds
   of the jpeg library have HAVE_STDLIB_H defined in the jconfig.h header file.
   Without the -qlanglvl=ansi, some older xlc versions (e.g., V7.0) still
   fail, but newer xlc versions (e.g., V9.0) pass.  AKC - 2010/02/17

o  When building on Linux/UNIX platforms, the szip shared library files must
   be in the system library path.  This can be done by adding a link to
   the libsz.* files in the /usr/lib folder or by adding the library
   location to the LD_LIBRARY_PATH environment variable.
      Ex. export LD_LIBRARY_PATH=path_to_szip_lib:$LD_LIBRARY_PATH
   Optionally, one can use the static szip library files by adding '-static'
   to the CFLAGS environment variable.

o  Existing data written by an HDF4 Library prior to HDF 4.2r2:
   When a one-dimensional SDS and a dimension scale have
   the same name, subsequent accesses to the dimension scale or to the
```

SDS might produce undesired results because the libraries could not distinguish between the two objects. In the case of writing, data might even be corrupted. For example, SDS data might be written to a dimension variable or vice versa. (bugzilla #624)

HDF4 Library Releases 4.2r2 and later make a distinction between an SDS and a dimension variable. However, as with older versions, these recent versions are unable to detect such conflicts in files created by earlier releases. It is therefore STRONGLY recommended to check for such name duplication before working with data created with a pre-4.2r2 library.

The functions SDgetnumvars_byname and SDnametoindices are provided to help detect such name conflicts and select the correct object to access, respectively; see the HDF Reference Manual entries for further details.
FB - 2009/01/26
BMR - revised 2011/06/24

- N-bit compression is not supported with Fortran APIs.

- Using both fill-value and compression on SD datasets does not work.

- When using PGI compilers, make sure that the JPEG library is also compiled with a PGI C compiler; linking with a JPEG library built with gcc causes JPEG library tests to fail. To bypass the problem:

    - Set LIBS flag to $PGI_JPEG_INSTALL_DIR/lib/libjpeg.a where $PGI_JPEG_INSTALL_DIR points to the installation directory for the PGI-compiled JPEG library:

      ```
      setenv LIBS $PGI_JPEG_INSTALL_DIR/lib/libjpeg.a
      ```

    - Use the --with-jpeg=$PGI_JPEG_INSTALL_DIR configure flag to configure with the PGI-compiled JPEG library:

      ```
      ./configure --with-jpeg=$PGI_JPEG_INSTALL_DIR --with-zlib....
      ```

- In order for the API SDgetdatasize to get the correct compressed size of the data, the dataset needs to be closed (SDendaccess) or read

```
(SDreaddata) after being written and before SDgetdatasize is called.
BMR - 2008/11/22
```