HDF5 1.10.6

Release Information

Version	HDF5 1.10.6
Release Date	2019-12-23
Download	Download
Release Notes	Release Notes
Tested Platforms and Configuration Features	Platforms / Features
Compatibility Report	1.10.5 vs 1.10.6
Changes Between Releases	Software Changes From Release to Release for HDF5-1.10
New Features	Introduced in 1.10.6
Newsletter	Release Announcement

Files

File	Type	Comment	Install Instructions	Md5 Checksum
hdf5-1.10.6.tar.gz	Source release	Unix Gzipped source tar file. See *Methods to obtain* (below).	release_docs/ directory in source	hdf5-1.10.6.md5
hdf5-1.10.6.tar.bz2	Source release	Unix Bzipped source tar file	release_docs/ directory in source	"
hdf5-1.10.6.zip	Source release	Windows zip file	release_docs/ directory in source	"
CMake-hdf5-1.10.6.tar.gz	CMake source release	File to build HDF5 with CMake (Unix). See *Methods to obtain* (below).	Building HDF5 with CMake	"
CMake-hdf5-1.10.6.zip	CMake source release	File to build HDF5 with CMake on Windows	Building HDF5 with CMake	"
Ready to use Binaries	Binary File	Pre-built binary distributions for Unix and Windows		
HDF5 Plugins	Binary File	Pre-built HDF5 plugin distributions for Unix and Windows	Information on HDF5 Filter Plugins	

Methods to obtain (gz file)

- firefox – Downoad file and then run: `gzip <distribution>.tar.gz | tar xzf -`
- chrome – Download file and then run: `gzip -cd <distribution>.tar.gz | tar xvf -`
- wget – `wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.N/hdf5-1.N.N/src/<distribution>.tar.gz`

```
gzip -cd <distribution>.tar.gz | tar xvf -
```

Tested Platforms and Configuration Features

NOTE: HDF5-1.10 *requires* MPI 3.

```
Supported Platforms
===================

    Linux 2.6.32-696.20.1.el6.ppc64 gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-23)
    #1 SMP ppc64 GNU/Linux        g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-23)
    (ostrich)                     GNU Fortran (GCC) 4.4.7 20120313 (Red Hat
4.4.7-23)
                                  IBM XL C/C++ V13.1
                                  IBM XL Fortran V15.1

    Linux 3.10.0-327.10.1.el7   GNU C (gcc), Fortran (gfortran), C++ (g++)
    #1 SMP x86_64 GNU/Linux       compilers:
    (jelly/kituo/moohan)          Version 4.8.5 20150623 (Red Hat 4.8.5-4)
                                    Version 4.9.3, Version 5.2.0
                                  Intel(R) C (icc), C++ (icpc), Fortran (icc)
                                  compilers:
                                     Version 17.0.0.098 Build 20160721
                                  MPICH 3.1.4 compiled with GCC 4.9.3

    SunOS 5.11 32- and 64-bit     Sun C 5.12 SunOS_sparc
    (emu)                         Sun Fortran 95 8.6 SunOS_sparc
                                  Sun C++ 5.12 SunOS_sparc

    Windows 7                     Visual Studio 2015 w/ Intel Fortran 18 (cmake)

    Windows 7 x64                 Visual Studio 2015 w/ Intel C, Fortran 2018
(cmake)
                                  Visual Studio 2015 w/ MSMPI 8 (cmake)

    Windows 10                    Visual Studio 2015 w/ Intel Fortran 18 (cmake)

    Windows 10 x64                Visual Studio 2015 w/ Intel Fortran 18 (cmake)
                                  Visual Studio 2017 w/ Intel Fortran 19 (cmake)
                                  Visual Studio 2019 w/ Intel Fortran 19 (cmake)

    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/icpc/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/icpc/ifort version 19.0.4


Tested Configuration Features Summary
=====================================

    In the tables below
          y   = tested
          n   = not tested in this release
          C   = Cluster
          W   = Workstation
          x   = not working in this release
          dna = does not apply
          ( ) = footnote appears below second table
```

<blank> = testing incomplete on this feature or platform

Platform	C parallel	F90/ F2003	F90 parallel	C++	zlib	SZIP
Solaris2.11 32-bit	n	y/y	n	y	y	y
Solaris2.11 64-bit	n	y/n	n	y	y	y
Windows 7	y	y/y	n	y	y	y
Windows 7 x64	y	y/y	y	y	y	y
Windows 7 Cygwin	n	y/n	n	y	y	y
Windows 7 x64 Cygwin	n	y/n	n	y	y	y
Windows 10	y	y/y	n	y	y	y
Windows 10 x64	y	y/y	n	y	y	y
Mac OS X Yosemite 10.10.5 64-bit	n	y/y	n	y	y	y
Mac OS X El Capitan 10.11.6 64-bit	n	y/y	n	y	y	y
MacOS High Sierra 10.13.6 64-bit	n	y/y	n	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 PGI	n	y/y	n	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 GNU	y	y/y	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 Intel	n	y/y	n	y	y	y
Linux 2.6.32-573.18.1.el6.ppc64	n	y/y	n	y	y	y

Platform	Shared C libs	Shared F90 libs	Shared C++ libs	Thread-safe
Solaris2.11 32-bit	y	y	y	y
Solaris2.11 64-bit	y	y	y	y
Windows 7	y	y	y	y
Windows 7 x64	y	y	y	y
Windows 7 Cygwin	n	n	n	y
Windows 7 x64 Cygwin	n	n	n	y
Windows 10	y	y	y	y
Windows 10 x64	y	y	y	y
Mac OS X Yosemite 10.10.5 64-bit	y	y	y	y
Mac OS X El Capitan 10.11.6 64-bit	y	y	y	y
MacOS High Sierra 10.13.6 64-bit	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 PGI	y	y	y	n
CentOS 7.2 Linux 3.10.0 x86_64 GNU	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 Intel	y	y	y	n
Linux 2.6.32-573.18.1.el6.ppc64	y	y	y	n

Compiler versions for each platform are listed in the preceding "Supported Platforms" table.

More Tested Platforms

The following configurations are not supported but have been tested for this release.

Linux 2.6.32-754.11.1.el6 #1 SMP x86_64 GNU/Linux (mayll/platypus)	GNU C (gcc), Fortran (gfortran), C++ (g++) compilers: Version 4.4.7 20120313 Version 4.9.3, 5.3.0, 6.2.0 PGI C, Fortran, C++ for 64-bit target on x86-64; Version 17.10-0 Intel(R) C (icc), C++ (icpc), Fortran (icc) compilers: Version 17.0.4.196 Build 20170411 MPICH 3.1.4 compiled with GCC 4.9.3

```
    Linux 3.10.0-327.18.2.el7     GNU C (gcc) and C++ (g++) compilers
    #1 SMP x86_64 GNU/Linux          Version 4.8.5 20150623 (Red Hat 4.8.5-4)
    (jelly)                       with NAG Fortran Compiler Release 6.2(Chiyoda)
                                  GCC Version 7.1.0
                                  MPICH 3.2-GCC-4.9.3
                                  MPICH 3.2.1-GCC-7.2.0-2.29
                                  OpenMPI 2.1.5-GCC-7.2.0-2.29
                                  Intel(R) C (icc) and C++ (icpc) compilers
                                     Version 17.0.0.098 Build 20160721
                                  with NAG Fortran Compiler Release 6.2(Chiyoda)

    Linux 3.10.0-327.10.1.el7     MPICH 3.2 compiled with GCC 5.3.0
    #1 SMP x86_64 GNU/Linux
    (moohan)

    Linux 2.6.32-573.18.1.el6.ppc64  MPICH mpich 3.1.4 compiled with
    #1 SMP ppc64 GNU/Linux           IBM XL C/C++ for Linux, V13.1
    (ostrich)                        and IBM XL Fortran for Linux, V15.1

    Fedora30 5.3.11-200.fc30.x86_64
    #1 SMP x86_64  GNU/Linux      GNU 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)

    Mac OS X 10.11.6, Darwin,     Apple clang version 7.3.0 from Xcode 7.3
    15.6.0, x86-64                gfortran GNU Fortran (GCC) 5.2.0
    (osx1011test)                 Intel icc/icpc/ifort version 16.0.2

    macOS 10.12.6, Darwin,        Apple clang LLVM version 8.1.0 from Xcode 8.3
    16.6.0, x86_64                gfortran GNU Fortran (GCC) 7.1.0
    (kite)                        Intel icc/icpc/ifort version 17.0.2
```

```
    Windows 7 x64                Visual Studio 2008
```

Release Notes

```
HDF5 version 1.10.6 released on 2019-12-23
================================================================================


INTRODUCTION

This document describes the differences between this release and the previous
HDF5 release. It contains information on the platforms tested and known
problems in this release. For more details check the HISTORY*.txt files in the
HDF5 source.

Note that documentation in the links below will be updated at the time of each
final release.

Links to HDF5 documentation can be found on The HDF5 web page:

     https://portal.hdfgroup.org/display/HDF5/HDF5

The official HDF5 releases can be obtained from:

     https://www.hdfgroup.org/downloads/hdf5/

Changes from Release to Release and New Features in the HDF5-1.10.x release series
can be found at:

     https://portal.hdfgroup.org/display/HDF5/HDF5+Application+Developer%27s+Guide

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

     help@hdfgroup.org


CONTENTS

- New Features
- Support for new platforms and languages
- Bug Fixes since HDF5-1.10.5
- Supported Platforms
- Tested Configuration Features Summary
- More Tested Platforms
- Known Problems
- CMake vs. Autotools installations


New Features
============

    Configuration:
    -------------
```

- 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, HDFFV-10962)

- Update CMake options to match new autotools options

  Add configure options (autotools - CMake):
        enable-asserts   HDF5_ENABLE_ASSERTS
        enable-symbols   HDF5_ENABLE_SYMBOLS
        enable-profiling   HDF5_ENABLE_PROFILING
        enable-optimization HDF5_ENABLE_OPTIMIZATION
  In addition NDEBUG is no longer forced defined and relies on the CMake
  process.

  (ADB - 2019/10/07, HDFFV-100901, HDFFV-10637, TRILAB-97)

- 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 have been a number of mingw issues that have 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 have 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)

    - Add options to enable or disable building tools and tests

      Configure options --enable-tests and --enable-tools were added for
      autotools configure.  These options are enabled by default, and can be
      disabled with either --disable-tests (or tools) or --enable-tests=no
      (or --enable-tools=no).  Build time is reduced ~20% when tools are
      disabled, 35% when tests are disabled, 45% when both are disabled.
      Re-enabling them after the initial build requires running configure
      again with the option(s) enabled.

      (LRK - 2019/06/12, HDFFV-9976)

    - Change tools tests to search the error stack

      There are some use cases which can cause the error stack of tools to be
      different then the expected output. These tests now use grepTest.cmake;
      this was changed to allow the error file to be searched for an expected
string.

      (ADB - 2019/04/15, HDFFV-10741)


    Library:
    --------
    - Added S3 and HDFS Virtual File Drivers (VFDs) to HDF5

      These new VFDs have been introduced in HDF5-1.10.6.  Instructions to
      enable them when configuring HDF5 on Linux and Mac may be found at
      https://portal.hdfgroup.org/display/HDF5/Virtual+File+Drivers+-+S3+and+HDFS.

      Installing on Windows requires CMake 3.13 and the following additional setup.
            Install openssl library (with dev files);
            from "Shining Light Productions". msi package preferred.

            PATH should have been updated with the installation dir.
            set ENV variable OPENSSL_ROOT_DIR to the installation dir.
            set ENV variable OPENSSL_CONF to the cfg file, likely
%OPENSSL_ROOT_DIR%\bin\openssl.cfg
            Install libcurl library (with dev files);
            download the latest released version using git:
https://github.com/curl/curl.git

            Open a Visual Studio Command prompt
            change to the libcurl root folder
            run the "buildconf.bat" batch file
            change to the winbuild directory
            nmake /f Makefile.vc mode=dll MACHINE=x64
            copy libcurl-vc-x64-release-dll-ipv6-sspi-winssl dir to C:\curl
(installation dir)
```

```
            set ENV variable CURL_ROOT to C:\curl (installation dir)
            update PATH ENV variable to %CURL_ROOT%\bin (installation bin dir).
            the aws credentials file should be in %USERPROFILE%\.aws folder
            set the ENV variable
"HDF5_ROS3_TEST_BUCKET_URL=https://s3.us-east-2.amazonaws.com/hdf5ros3"

      (ADB - 2019/09/12, HDFFV-10854)

    C++ Library:
    ------------
    - Added new wrappers for H5Pset/get_create_intermediate_group()
        LinkCreatPropList::setCreateIntermediateGroup()
        LinkCreatPropList::getCreateIntermediateGroup()

      (BMR - 2019/04/22, HDFFV-10622)


    Java Library:
    ----------------
    - Fixed a failure in JUnit-TestH5P on 32-bit architectures

      (JTH - 2019/04/30)


Support for new platforms, languages 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


Bug Fixes since HDF5-1.10.5 release
==================================

    Library
    -------
    - Improved performance when creating a large number of small datasets by
      retrieving default property values from the API context instead of doing
      skip list searches. More work is required to achieve parity with HDF5 1.8.

      (CJH - 2019/12/10, HDFFV-10658)

    - Fixed user-created data access properties not existing in the property list
      returned by H5Dget_access_plist. Thanks to Steven Varga for submitting a
      reproducer and a patch.

      (CJH - 2019/12/9, HDFFV-10934)

    - Inappropriate linking with deprecated MPI C++ libraries

      HDF5 does not define *_SKIP_MPICXX in the public headers, so applications
      can inadvertently wind up linking to the deprecated MPI C++ wrappers.

      MPICH_SKIP_MPICXX and OMPI_SKIP_MPICXX have both been defined in H5public.h
      so this should no longer be an issue. HDF5 makes no use of the deprecated
      MPI C++ wrappers.
```

(DER - 2019/09/17, HDFFV-10893)

- fcntl(2)-based file locking incorrectly passed the lock argument struct instead of a pointer to the struct, causing errors on systems where flock(2) is not available.

  File locking is used when files are opened to enforce SWMR semantics. A lock operation takes place on all file opens unless the HDF5_USE_FILE_LOCKING environment variable is set to the string "FALSE". flock(2) is preferentially used, with fcntl(2) locks as a backup if flock(2) is unavailable on a system (if neither is available, the lock operation fails). On these systems, the file lock will often fail, which causes HDF5 to not open the file and report an error.

  This bug only affects POSIX systems. Win32 builds on Windows use a no-op locking call which always succeeds. Systems which exhibit this bug will have H5_HAVE_FCNTL defined but not H5_HAVE_FLOCK in the configure output.

  This bug affects HDF5 1.10.0 through 1.10.5.

  fcntl(2)-based file locking now correctly passes the struct pointer.

  (DER - 2019/08/27, HDFFV-10892)

- Fixed a bug caused by a bad tag value when condensing object header messages

  There was an assertion failure when moving messages from running a user test program with library release HDF5 1.10.4. It was because the tag value (object header's address) was not set up when entering the library routine H5O__chunk_update_idx(), which eventually verifies the metadata tag value when protecting the object header.

  The problem was fixed by replacing FUNC_ENTER_PACKAGE in H5O__chunk_update_idx()
  with FUNC_ENTER_PACKAGE_TAG(oh->cache_info.addr) to set up the metadata tag.

  (VC - 2019/08/23, HDFFV-10873)

- Fixed the test failure from test_metadata_read_retry_info() in test/swmr.c

  The test failure is due to an incorrect number of bins returned for retry info (info.nbins). The # of bins expected for 101 read attempts is 3 instead of 2. The routine H5F_set_retries() in src/H5Fint.c calculates the # of bins by first obtaining the log10 value for (read attempts - 1). For PGI/19, the log10 value for 100 read attempts is 1.9999999999999998 instead of 2.00000. When casting the log10 value to unsigned later on, the decimal part is chopped off causing the test failure.

  This was fixed by obtaining the rounded integer value (HDceil) for the log10 value of read attempts first before casting the result to unsigned.

  (VC - 2019/8/14, HDFFV-10813)

- Fixed an issue when creating a file with non-default file space info together with library high bound setting to H5F_LIBVER_V18.

When setting non-default file space info in fcpl via
H5Pset_file_space_strategy() and then creating a file with both high and
low library bounds set to H5F_LIBVER_V18 in fapl, the library succeeds in
creating the file. File creation should fail because the feature of
setting non-default file space info does not exist in library release 1.8
or earlier.

This was fixed by setting and checking the proper version in the file
space info message based on the library low and high bounds when creating
and opening the HDF5 file.

(VC - 2019/6/25, HDFFV-10808)

- Fixed an issue where copying a version 1.8 dataset between files using
H5Ocopy fails due to an incompatible fill version

When using the HDF5 1.10.x H5Ocopy() API call to copy a version 1.8
dataset to a file created with both high and low library bounds set to
H5F_LIBVER_V18, the H5Ocopy() call will fail with the error stack indicating
that the fill value version is out of bounds.

This was fixed by changing the fill value message version to
H5O_FILL_VERSION_3
(from H5O_FILL_VERSION_2) for H5F_LIBVER_V18.

(VC - 2019/6/14, HDFFV-10800)

- Fixed a bug that would cause an error or cause fill values to be
incorrectly read from a chunked dataset using the "single chunk" index if
the data was held in cache and there was no data on disk.

(NAF - 2019/03/06)

- Fixed a bug that could cause an error or cause fill values to be
incorrectly read from a dataset that was written to using H5Dwrite_chunk
if the dataset was not closed after writing.

(NAF - 2019/03/06, HDFFV-10716)

- Fixed memory leak in scale offset filter

In a special case where the MinBits is the same as the number of bits in
the datatype's precision, the filter's data buffer was not freed, causing
the memory usage to grow. In general the buffer was freed correctly. The
Minbits are the minimal number of bits to store the data values. Please
see the reference manual for H5Pset_scaleoffset for the details.

(RL - 2019/3/4, HDFFV-10705)

Configuration

- Correct option for default API version

CMake options for default API version are not mutually exclusive.
Change the multiple BOOL options to a single STRING option with the
strings; v16, v18, v110.

(ADB - 2019/08/12, HDFFV-10879)

```
    Tools
    -----
    - h5repack was fixed to repack datasets with external storage
      to other types of storage.

      New test added to repack files and verify the correct data using h5diff.

      (JS - 2019/09/25, HDFFV-10408)
      (ADB - 2019/10/02, HDFFV-10918)


Supported Platforms
===================

    Linux 2.6.32-696.20.1.el6.ppc64 gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-23)
    #1 SMP ppc64 GNU/Linux        g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-23)
    (ostrich)                     GNU Fortran (GCC) 4.4.7 20120313 (Red Hat
4.4.7-23)
                                  IBM XL C/C++ V13.1
                                  IBM XL Fortran V15.1

    Linux 3.10.0-327.10.1.el7   GNU C (gcc), Fortran (gfortran), C++ (g++)
    #1 SMP x86_64 GNU/Linux       compilers:
    (jelly/kituo/moohan)          Version 4.8.5 20150623 (Red Hat 4.8.5-4)
                                    Version 4.9.3, Version 5.2.0
                                  Intel(R) C (icc), C++ (icpc), Fortran (icc)
                                  compilers:
                                     Version 17.0.0.098 Build 20160721
                                  MPICH 3.1.4 compiled with GCC 4.9.3

    SunOS 5.11 32- and 64-bit     Sun C 5.12 SunOS_sparc
    (emu)                         Sun Fortran 95 8.6 SunOS_sparc
                                  Sun C++ 5.12 SunOS_sparc

    Windows 7                     Visual Studio 2015 w/ Intel Fortran 18 (cmake)

    Windows 7 x64                 Visual Studio 2015 w/ Intel C, Fortran 2018
(cmake)
                                  Visual Studio 2015 w/ MSMPI 8 (cmake)

    Windows 10                    Visual Studio 2015 w/ Intel Fortran 18 (cmake)

    Windows 10 x64                Visual Studio 2015 w/ Intel Fortran 18 (cmake)
                                  Visual Studio 2017 w/ Intel Fortran 19 (cmake)
                                  Visual Studio 2019 w/ Intel Fortran 19 (cmake)

    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/icpc/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/icpc/ifort version 19.0.4


Tested Configuration Features Summary
=====================================
```

In the tables below
y = tested
n = not tested in this release
C = Cluster
W = Workstation
x = not working in this release
dna = does not apply
( ) = footnote appears below second table
<blank> = testing incomplete on this feature or platform

Platform	C parallel	F90/ F2003	F90 parallel	C++	zlib	SZIP
Solaris2.11 32-bit	n	y/y	n	y	y	y
Solaris2.11 64-bit	n	y/n	n	y	y	y
Windows 7	y	y/y	n	y	y	y
Windows 7 x64	y	y/y	y	y	y	y
Windows 7 Cygwin	n	y/n	n	y	y	y
Windows 7 x64 Cygwin	n	y/n	n	y	y	y
Windows 10	y	y/y	n	y	y	y
Windows 10 x64	y	y/y	n	y	y	y
Mac OS X Yosemite 10.10.5 64-bit	n	y/y	n	y	y	y
Mac OS X El Capitan 10.11.6 64-bit	n	y/y	n	y	y	y
MacOS High Sierra 10.13.6 64-bit	n	y/y	n	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 PGI	n	y/y	n	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 GNU	y	y/y	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 Intel	n	y/y	n	y	y	y
Linux 2.6.32-573.18.1.el6.ppc64	n	y/y	n	y	y	y

Platform	Shared C libs	Shared F90 libs	Shared C++ libs	Thread-safe
Solaris2.11 32-bit	y	y	y	y
Solaris2.11 64-bit	y	y	y	y
Windows 7	y	y	y	y
Windows 7 x64	y	y	y	y
Windows 7 Cygwin	n	n	n	y
Windows 7 x64 Cygwin	n	n	n	y
Windows 10	y	y	y	y
Windows 10 x64	y	y	y	y
Mac OS X Yosemite 10.10.5 64-bit	y	y	y	y
Mac OS X El Capitan 10.11.6 64-bit	y	y	y	y
MacOS High Sierra 10.13.6 64-bit	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 PGI	y	y	y	n
CentOS 7.2 Linux 3.10.0 x86_64 GNU	y	y	y	y
CentOS 7.2 Linux 3.10.0 x86_64 Intel	y	y	y	n
Linux 2.6.32-573.18.1.el6.ppc64	y	y	y	n

Compiler versions for each platform are listed in the preceding "Supported Platforms" table.

More Tested Platforms

The following configurations are not supported but have been tested for this release.

Linux 2.6.32-754.11.1.el6 #1 SMP x86_64 GNU/Linux (mayll/platypus)	GNU C (gcc), Fortran (gfortran), C++ (g++) compilers: Version 4.4.7 20120313

```
                                   Version 4.9.3, 5.3.0, 6.2.0
                                PGI C, Fortran, C++ for 64-bit target on
                                x86-64;
                                   Version 17.10-0
                                Intel(R) C (icc), C++ (icpc), Fortran (icc)
                                compilers:
                                   Version 17.0.4.196 Build 20170411
                                MPICH 3.1.4 compiled with GCC 4.9.3

    Linux 3.10.0-327.18.2.el7   GNU C (gcc) and C++ (g++) compilers
    #1 SMP x86_64 GNU/Linux        Version 4.8.5 20150623 (Red Hat 4.8.5-4)
    (jelly)                     with NAG Fortran Compiler Release 6.2(Chiyoda)
                                GCC Version 7.1.0
                                MPICH 3.2-GCC-4.9.3
                                MPICH 3.2.1-GCC-7.2.0-2.29
                                OpenMPI 2.1.5-GCC-7.2.0-2.29
                                Intel(R) C (icc) and C++ (icpc) compilers
                                   Version 17.0.0.098 Build 20160721
                                with NAG Fortran Compiler Release 6.2(Chiyoda)

    Linux 3.10.0-327.10.1.el7   MPICH 3.2 compiled with GCC 5.3.0
    #1 SMP x86_64 GNU/Linux
    (moohan)

    Linux 2.6.32-573.18.1.el6.ppc64  MPICH mpich 3.1.4 compiled with
    #1 SMP ppc64 GNU/Linux           IBM XL C/C++ for Linux, V13.1
    (ostrich)                        and IBM XL Fortran for Linux, V15.1

    Fedora30 5.3.11-200.fc30.x86_64
    #1 SMP x86_64  GNU/Linux    GNU 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)

    Mac OS X 10.11.6, Darwin,   Apple clang version 7.3.0 from Xcode 7.3
    15.6.0, x86-64              gfortran GNU Fortran (GCC) 5.2.0
    (osx1011test)               Intel icc/icpc/ifort version 16.0.2

    macOS 10.12.6, Darwin,      Apple clang LLVM version 8.1.0 from Xcode 8.3
    16.6.0, x86_64              gfortran GNU Fortran (GCC) 7.1.0
    (kite)                      Intel icc/icpc/ifort version 17.0.2

    Windows 7 x64               Visual Studio 2008


Known Problems
==============
    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

    At present, metadata cache images may not be generated by parallel
    applications.  Parallel applications can read files with metadata cache
    images, but since this is a collective operation, a deadlock is possible
    if one or more processes do not participate.

    Three tests fail with OpenMPI 3.0.0/GCC-7.2.0-2.29:
```

```
        testphdf5 (ecdsetw, selnone, cchunk1, cchunk3, cchunk4, and actualio)
        t_shapesame (sscontig2)
        t_pflush1/fails on exit
    The first two tests fail attempting collective writes.

    Parallel builds using OpenMPI 3.03 or later and romio fail several tests
    with collective writes or compression that will not fail when ompio is used
    instead of romio.  This can be done by adding "--mca io ompio" to the mpirun
    command.  For example, in autotools builds RUNPARALLEL can be set to
    "mpirun --mca io ompio -n 6" provided ompio is installed.

    CPP ptable test fails on VS2017 with Intel compiler, JIRA issue: HDFFV-10628.
    This test will pass with VS2015 with Intel compiler.

    Older MPI libraries such as OpenMPI 2.0.1 and MPICH 2.1.5 were tested
    while attempting to resolve the Jira issue: HDFFV-10540.
    The known problems of reading or writing > 2GBs when using MPI-2 was
    partially resolved with the MPICH library. The proposed support recognizes
    IO operations > 2GB and if the datatype is not a derived type, the library
    breaks the IO into chunks which can be input or output with the existing
    MPI 2 limitations, i.e. size reporting and function API size/count
    arguments are restricted to be 32 bit integers.  For derived types larger
    than 2GB, MPICH 2.1.5 fails while attempting to read or write data.
    OpenMPI in contrast, implements MPI-3 APIs even in the older releases
    and thus does not suffer from the 32 bit size limitation described here.
    OpenMPI releases prior to v3.1.3 appear to have other datatype issues however,
    e.g. within a single parallel test (testphdf5) the subtests (cdsetr, eidsetr)
    report data verfication errors before eventually aborting.
    The most recent versions of OpenMPI (v3.1.3 or newer) have evidently
    resolved these isses and parallel HDF5 testing does not currently report
    errors though occasional hangs have been observed.

    Known problems in previous releases can be found in the HISTORY*.txt files
    in the HDF5 source. Please report any new problems found to
    help@hdfgroup.org.


CMake vs. Autotools installations
=================================
While both build systems produce similar results, there are differences.
Each system produces the same set of folders on linux (only CMake works
on standard Windows); bin, include, lib and share. Autotools places the
COPYING and RELEASE.txt file in the root folder, CMake places them in
the share folder.

The bin folder contains the tools and the build scripts. Additionally, CMake
creates dynamic versions of the tools with the suffix "-shared". Autotools
installs one set of tools depending on the "--enable-shared" configuration
option.
  build scripts
  -------------
  Autotools: h5c++, h5cc, h5fc
  CMake: h5c++, h5cc, h5hlc++, h5hlcc

The include folder holds the header files and the fortran mod files. CMake
places the fortran mod files into separate shared and static subfolders,
while Autotools places one set of mod files into the include folder. Because
CMake produces a tools library, the header files for tools will appear in
the include folder.
```

The lib folder contains the library files, and CMake adds the pkgconfig subfolder with the hdf5*.pc files used by the bin/build scripts created by the CMake build. CMake separates the C interface code from the fortran code by creating C-stub libraries for each Fortran library. In addition, only CMake installs the tools library. The names of the szip libraries are different between the build systems.

The share folder will have the most differences because CMake builds include a number of CMake specific files for support of CMake's find_package and support

```
for the HDF5 Examples CMake project.
```

Compatibility Report