# Building HDF5 with CMake

## Overview

A CTest script and platform configuration file can be used to simplify building with CMake. The following instructions are provided for this purpose.

If you need to build with more complex options, you may prefer to follow the instructions in the release_docs/ directory of the HDF5 source code.



## Preconditions

- The CMake HDF5 source release file (for either Unix or Windows) **MUST** have been downloaded.

  You can obtain the CMake HDF5 source file by either of these ways:

  a. Select a specific HDF5 release from the **Downloads** page on the Support Portal (scroll down to see the releases). On the specific release page see the table under "Files". Select either the CMake-hdf5-N.N.N.tar.gz or CMake-hdf5.N.N.N.zip file. ***NOTE: Users should NOT use 1.10 releases prior to HDF5-1.10.3.*** For more information see the Software Changes from Release to Release for HDF5-1.10 page.

  b. Go to the **latest HDF5 source code** on The HDF Group web site. (Scroll down to see *Cmake Versions*.)

- CMake **MUST** be installed. The configuration scripts require a minimum CMake version 3.12, although version 3.15 is recommended.
- Blank spaces **MUST NOT** be used in directory path names as this will cause the build to fail.
- (Optional) On Windows, an NSIS or WiX should be installed in order to create an install image with CPack. NSIS will create a .exe installer. WiX will create a .msi installer.

## Quick Instructions

The quick instructions for building HDF5 with CMake are:

1. Download the CMake source code (CMake-hdf5-N.N.N.tar.gz or CMake-hdf5.N.N.N.zip).
2. Uncompress it.
3. From the command line, go in to the top CMake-hdf5-N.N.N directory and execute the build script (`build*`) for your platform.

If you encounter any issues, then see the instructions below.

## Build Instructions

1. Create a working directory. For HPC systems the working directory should be in a scratch or parallel file system space, since testing will use this space.

2. Uncompress the HDF5 CMake source code file into the working directory. It will contain a `CMake-hdf5-N` directory (where N is the release version). ( See **Preconditions** for how to obtain the software )

3. From the ***command line***, go into the CMake-hdf5-N directory, which contains:

| | |
|---|---|
| **build*.sh (.bat)** | Build Script(s) |
| **CTestScript.cmake** | ctest Command |
| **hdf5-N** | HDF5 Source Code |
| **HDF5config.cmake** | Configuration File |
| **HDF5options.cmake** | User modifiable Options |
| **SZip.tar.gz** | External Library for SZIP Compression |

| | |
|---|---|
| **ZLib.tar.gz** | External Library for ZLIB Compression |

4. By default, HDF5 will be built:

    - Without Fortran
    - With SZIP compression enabled
    - With ZLIB compression enabled
    - In Release Mode
    - With shared libraries

    Users can change the options that HDF5 is built with by adding options to the build command (see the batch files/test script below) or by modifying the HDF5options.cmake file. The HDF5options.cmake file will override any options set in the configuration file. For more information see the How to Change HDF5 CMake Build Options page.

    REQUIRED: Visual Studio Express users must change the build options to turn off packaging or the build will fail.

5. Execute the batch file or shell script for your platform. It contains the `ctest` command that you need to run to build HDF5. (See Troubleshooting if you do not see your platform).

    Below are example build scripts that you may find:

| Platform | Batch File or Shell Script | Contains following `ctest` command |
|---|---|---|
| Windows 64-bit VS 2015 | build-VS2015-64.bat | ctest -S HDF5config.cmake,BUILD_GENERATOR=VS201564 -C Release -V -O hdf5.log |
| Windows 32-bit VS 2015 | build-VS2015-32.bat | ctest -S HDF5config.cmake,BUILD_GENERATOR=VS2015 -C Release -V -O hdf5.log |
| Unix | build-unix.sh | ctest -S HDF5config.cmake,BUILD_GENERATOR=Unix -C Release -V -O hdf5.log |
| Unix (HPC) | build-unix-hpc.sh | ctest -S HDF5config.cmake,HPC=sbatch,MPI=true,BUILD_GENERATOR=Unix -C Release -V -O hdf5.log |

    Where the `ctest` command is using these options:

    - The `-S` option uses the script version of ctest. `HDF5config.cmake` is the configuration file.
    - The `-C` option specifies the build configuration which matches CTEST_BUILD_CONFIGURATION in the configuration file.
    - The `-V` option indicates verbose. Using the `-VV` option indicates *more* verbose. If encountering problems, **specify -VV for more verbose output.**
    - The `-O` option saves the output to a log file, *hdf5.log.*

6. Locate the built binary.

    The built binary will be in the build directory and will also be copied to the CMake-hdf5-N directory if successful. It will have the format:

    ```
    HDF5-N-<platform>.<zip or tar.gz>
    ```

    On Unix, <platform> will be "Linux". A similar `.sh` file will also be created.

    On Windows, <platform> will be "win64" or "win32". If you have an installer on your system, you will also see a similar file that ends in either `.exe` (NSIS) or `.msi` (WiX).

    If the built binary is not there, then see Troubleshooting for help.

7. Check what is included with your built binaries.

    You will find the libhdf5.settings file in the build directory. It contains information on how the binaries were built.

    If you uncompress the built binary, you will find the hdf5-config.cmake and hdf5-targets.cmake files (among others) in a cmakedi rectory. This cmake directory can be found in the same location as the lib, include, and bin directories on Windows and under sh are/ on Unix. The options in the hdf5-config.cmake file match those in the libhdf5.settings file.

    The binaries by default will include the static HDF5 C and C++ libraries, as well as the SZIP and ZLIB external libraries. Please NOTE that they will NOT include the HDF5 Fortran library. See the How to Change HDF5 CMake Build Options page for instructions on building with Fortran.

8. Follow the instructions below for compiling an application with the binaries that are built.

## Compiling an Application

## CMake Scripts for Building Applications

Simple scripts are provided for building applications with different languages and options. See CMake Scripts for Building Applications.

For a more complete script (and to help resolve issues) see the script provided with the HDF5 Examples below.

### HDF5 Examples

The build can be verified by compiling the HDF5 Examples included with the CMake source code. You can also use the solution files that are created with these examples, if building with Visual Studio. This can be helpful in diagnosing issues with how to compile your application.

A compressed file containing the examples, `HDF5Examples-*`, is provided in the source.

Uncompress the file to find the `HDF5Examples` directory. Go into the `HDF5Examples` directory and follow the instructions in `Using_CMake.txt` to build the examples.

In general, users must first set the HDF5_DIR environment variable to the installed location of the CMake configuration files for HDF5. For example, on Windows the following path might be set:

```
HDF5_DIR=C:/Program Files/HDF_Group/HDF5/1.N.N/cmake
```

You may look at `find_package` provided with the HDF5 Examples for how to compile an application. *Please be aware that `FindHDF5.cmake` is not provided by and cannot be fixed by The HDF Group.*

**Steps for building the HDF5 Examples**

1. *Build HDF5 with CMake using the instructions on this page.*
2. Uncompress the `HDF5Examples-1.N.N-Source.zip` examples zip file.
3. Set the HDF5_DIR environment variable to the location of the HDF5 built binaries cmake directory:

```
C:/<yourpath>/CMake-hdf5-1.10.5/CMake-hdf5-1.10.5/HDF5-1.10.5-win64/HDF5-1.10.5-win64/cmake
```

4. From the command line go into the examples directory:

```
C:\<yourpath>\CMake-hdf5-1.10.5\CMake-hdf5-1.10.5\HDF5Examples-1.12.4-Source\HDF5Examples
```

5. Create a `build` directory.
6. Type: `cmake -G "Visual Studio 12 Win64"`
`c:\<yourpath>\CMake-hdf5-1.10.5\CMake-hdf5-1.10.5\HDF5Examples-1.12.4-Source\HDF5Examples`
7. Type: `cmake --build . --config Release`

### Framework

A framework for building applications is provided below:

- Unix Framework
- Windows Framework

Uncompress the file and follow the instructions in the `USING_CMake_Scripts.txt` file for building an application.

PLEASE note that the `cacheinit.cmake` file is missing from the multi-file build files. However, you can simply use the version provided with the single file build. Specifically, copy `hdf5-app\app\config\cmake\cacheinit.cmake` to the directory `hdf5-app\mfapp\config\cmake\`.

An example of building a multi-file application in C in debug mode is provided here:

- Add your files to: `hdf5-app\mfapp\C\`
- Edit `hdf5-app\mfapp\C\C_sourcefiles.cmake` and list your files. For example:

```
set (hdf5app_name "myapp")

set (hdf5app

     ${PROJECT_SOURCE_DIR}/myfile1.c

     ${PROJECT_SOURCE_DIR}/myfile1.h
```

```
        ${PROJECT_SOURCE_DIR}/myfile2.c

        ${PROJECT_SOURCE_DIR}/myfile2.h

        ${PROJECT_SOURCE_DIR}/program.c

    )
```

- Run the `ctest` command for your platform. For example:

```
ctest -S
HDFconfig.cmake,BUILD_GENERATOR=VS201364,CTEST_SOURCE_NAME=mfapp,INSTALLDIR="C:/Program
Files/HDF_Group/HDF5/1.10.2" -C Debug -VV -O test.log
```

- Run your application. For a debug application, the executable will be here: `hdf5-app\build\bin\Debug\`

### Compile Scripts (h5cc, ..)

If building HDF5 on Linux with CMake, compile scripts (`h5cc`, `h5c++`, `...`) are created and can be found in the `bin/` directory of the installed binary. While similar to the compile scripts built with autotools (configure), they are not the same, but they can be used to compile a simple application. Please note that the `h5pcc` compile script does not get created when compiling Parallel HDF5 with CMake.

## Troubleshooting

**I can build HDF5 successfully but the findHDF5.cmake package does not populate HDF5_LIBRARIES. How do you use the HDF5 libraries that you built?**

The `FindHDF5.cmake` package is not supported or provided by The HDF Group. The HDF Group cannot change it. However, you can use `find_package.`

See the question below on how to use `find_package`.

**How do you use find_package with HDF5?**

To use `find_package` you will first need to make sure that HDF5_DIR is set correctly. For setting this environment variable see the Preconditions in the USING_HDF5_CMake.txt file in the source code.

See the CMakeLists.txt file provided with these examples for how to use find_package with HDF5.

Please note that the find_package invocation changed to require "shared" or "static":

```
FIND_PACKAGE(HDF5 COMPONENTS C HL NO_MODULE REQUIRED shared)
      FIND_PACKAGE(HDF5 COMPONENTS C HL NO_MODULE REQUIRED static)
```

Previously, the find_package invocation was:

```
FIND_PACKAGE(HDF5 COMPONENTS C HL NO_MODULE REQUIRED)
```

**My platform/compiler is not included. Can I still use the configuration files?**

Yes, you can but you will have to edit the HDF5config.cmake file and update the variable:

```
CTEST_CMAKE_GENERATOR
```

Other variables may be updated for informational purposes but are not required (for example, SITE_OS_BITS).

The generators for your platform can be seen by typing:

```
cmake --help
```

**What do I do if the build fails?**

I received an error during the build and the built and compressed HDF5 binary is not in the CMake-hdf5-N/build directory as I expected. How do I determine what the problem is?

If the error is not clear, then the first thing you may want to do is replace the -V (Dash Uppercase Vee) option for ctest in the build script to -VV (Dash Uppercase Vee Uppercase Vee). Then remove the build directory and re-run the build script. The output should be more verbose.

If the error is still not clear, then check the log files. You will find those in the build directory under CMake-hdf5-N. For example, on Unix the log files will be in:

`CMake-hdf5-N/build/Testing/Temporary/`

There are log files for the configure, test, and build.

**What do I do if I need to rebuild the software?**

If you have to rebuild HDF5, remove the build directory first.

**The library was built but there are no binaries. What do I do?**

To install or package the binaries, run either make install or cpack in the build/ directory.