Geant4 Installation Guide Documentation

Release 10.7

Geant4 Collaboration

Rev5.0 - December 4th, 2020

CONTENTS

There are several ways to install Geant4 on your computer either from binary packages or by compiling from scratch, and these are described below. Which one is available or best for you depends on both your operating system and usage requirements. In all cases, always use the most recent Geant4 release to ensure use of the latest bug fixes, features, and help the developers and community to provide quick user support.

CHAPTER

ONE

BUILD AND INSTALL GEANT4 FROM SOURCE

Geant4 uses CMake to configure a build system for compiling and installing the toolkit headers, libraries and support tools from scratch. To follow this method, please see *Geant4 System/Software Prerequisites* for the operating system and software requirements, followed by *Building and Installing from Source*.

Whilst every effort has been made to make this installation method robust and reliable, the multitude of platforms and system configurations mean we cannot guarantee that problems will not be encountered on platforms other than those listed in *Supported and Tested Platforms*.

In case of issues with building and installing Geant4, we welcome questions as well as feedback via our Discourse Forum. To help us deal with your problem as quickly as possible, please include as much detail as possible on the problem you have encountered. At minimum, you should let us know the platform and operating system version, C++ compiler type and version, CMake version, and any error messages. It also helps to list the sequence of commands you used so we can try and reproduce the issue.

If you feel you have found a genuine bug in the Geant4 CMake build, please report it to the CMake category on our Bugzilla. As with reports to Discourse, please include as much information as possible so that we can triage the bug and track it down quickly. We also welcome general feature requests and feedback on the system through both Discourse and Bugzilla.

CHAPTER

TWO

INSTALL GEANT4 VIA A PACKAGE MANAGER

Warning: These packages are not maintained by the Geant4 developers, but by helpful members of the community. Please go through each package manager's standard channels to report any installation issues or to request packaging of the latest release/patch. If you package Geant4 and would like to be added to the list below, please contact us via our Discourse Forum.

2.1 Spack on Linux/macOS

Spack 's Geant4 package may be installed with

```
$ spack install geant4
```

Spack allows different variants of Geant4 to be installed, and to see these run

```
$ spack info geant4
```

2.2 Homebrew on macOS/Linux

Homebrew's Geant4 formula may be installed with

```
$ brew install geant4
```

2.3 Conda on Linux/macOS

A Conda package for Geant4 is available via conda-forge and may be installed into an environment via

```
$ conda create -c conda-forge --name <my-environment> geant4
$ conda activate <my-environment>
```

Please see the associated feedstock for further information and support.

2.4 Macports

MacPorts supplies a port for Geant4 which may be installed with

```
$ sudo port install geant4
```

2.5 Linux System Package Managers

Several Linux distributions provide Geant4 through their system package managers, either directly or through community repositories. A partial list with links to details are available from Repology at:

- Packages named geant4
- Packages named geant (*but please note that the '4' in 'Geant4' is not part of the version!*)

2.6 LCG CVMFS Releases for CentOS7 and Ubuntu Linux

If your platform can use or has CVMFS installed, Geant4 is available through the LCG Releases from the `sft.cern.ch` repository for CentOS7. Ubuntu 18.04/20.04 packages are available in a handful of releases.

2.6.1 Geant4 System/Software Prerequisites

OS/Software Prerequisites

The following source/software *must* be present to either build/install Geant4 or develop applications using it:

- Geant4 Toolkit Source Code.
- C++ Compiler and Standard Library supporting the C++11 Standard:
  - Linux: GNU Compiler Collection 4.9.3 or higher.
    - It is strongly recommended to use the GCC compiler supplied by the package management system of your distribution unless this does not meet the minimum version requirement.
  - macOS: Apple Clang (Xcode) 11 or higher.
    - The command line tools must also be installed by running `xcode-select --install` from the terminal.
  - Windows: Visual Studio 2019, Community version or higher.

  The compiler and standard library need to support at least the following features of the C++11 Standard:
  - Template aliases, as defined in N2258.
  - Automatic type deduction, as defined in N1984.
  - Delegating constructors, as defined in N1986.
  - Enum forward declarations, as defined in N2764.
  - Explicit conversion operators, as defined in N2437.
  - Override control final keyword, as defined in N2928, N3206 and N3272.
  - Lambda functions, as defined in N2927.

  - Null pointer, as defined in N2431.
  - Override control override keyword, as defined in N2928, N3206 and N3272.
  - Range-based for, as defined in N2930.
  - Strongly typed enums, as defined in N2347.
  - Uniform initialisation, as defined in N2640.

- CMake 3.8 or higher.

  On Linux, we recommend that you use CMake as provided through the package management system of your distribution, unless it does not meet the minimum version requirement. In that case, we recommend you install it using the Linux binary installer for the latest version of CMake, available with instructions from the Kitware download site. This installer is highly portable and should work on the vast majority of distributions.

  On macOS and Windows, CMake is not installed by default, so we recommend that you install it using the most recent Darwin64 dmg (macOS) or Win32 exe (Windows) installers supplied by the Kitware download site. On macOS, you may also use the Homebrew or Macports package managers to install the required version.

  For more information on CMake, the CMake Help and Documentation should be consulted.

Supported and Tested Platforms

Geant4 is officially supported on the following operating system and compiler combinations:

- CentOS 7 Linux with GCC >=4.9.3, >=5.4, >=6.3, >=7.3, >= 8.2, >= 9.2, >= 10.2 64bit

  The minimum required versions of GCC may be installed on CentOS7 systems via the free Developer Toolset packages.

  Geant4 has been successfully compiled on other Linux distributions, including Debian, Ubuntu and openSUSE. The main requirement is that the system has a GCC of sufficient version to support C++11 installed. Please note that distributions other than CentOS are not officially supported. However, feedback and patches for non-CentOS platforms are welcome!

- macOS 11.0 (Big Sur), 10.15 (Catalina), 10.14 (Mojave), with Apple-LLVM (Xcode) 11, 12, 64bit.

  Big Sur support is preliminary, as is support for the Apple M1 chip

- Windows 10 with Visual Studio 2019, 32/64bit.

  There is currently no official support for building/using Geant4 through Windows Subsystem for Linux, but we welcome reports or feedback on use of Geant4 under this system via our Discourse Forum.

The following platforms and compilers are also tested but not officially supported

- CentOS 7 Linux with Intel C/C++ Compiler >=19.X. *Note that the Intel Compiler must be set up to use C++ headers and standard library supplied by GNU GCC >= 4.9 only to provide the required compatibility with the C++11 Standard.*
- CentOS 7 Linux with LLVM/Clang 7, 8, and 9
- CentOS 8 Linux with GCC 8 and 10
- Ubuntu Linux 18.04LTS with GCC 7 (system compiler)
- Ubuntu Linux 20.04LTS with GCC 9 (system compiler)

Prerequisites for Optional Components of Geant4

Geant4 has several optional components which if enabled require further software to be preinstalled on your system. These components and their requirements are listed below.

On Linux, we strongly recommend that you install these through the package management system of your distribution unless these do not meet the version and (for C++ packages) standard requirements listed. You should consult the documentation of your distribution for information on the packages that provide the needed software libraries and headers.

On macOS and Windows, we strongly recommend installing any required packages through binary dmg/exe installers supplied by the vendors of the packages. Installation and use of packages on macOS through Homebrew or MacPorts is not tested, but you may build Geant4 using packages installed through these systems with that caveat.

CLHEP, Expat, and zlib Support Libraries

Geant4 distributes minimal versions of the CLHEP , Expat, and zlib sources with the toolkit to help cross-platform usage.

These internal versions are built and installed by default (*except for Expat on Linux and macOS Platforms*), but Geant4 can be configured to use existing installs of these packages if required (see *Geant4 Build Options* for details). If existing installs are used, they must meet the following version/standard requirements:

- CLHEP: 2.4.4.0 or higher, compiled against the same C++ Standard as Geant4 (C++11 by default)
- Expat: 2.0.1 or higher
- zlib: 1.2.3 or higher

GDML XML Geometry Support

To enable use of geometry reading/writing from GDML XML files, the Xerces-C++ headers and library >=3 must be installed, compiled against the same C++ Standard as Geant4 (C++11 by default). On Unix systems, it should also be configured and built with `netaccessor-curl`, and the used `libcurl` should support SSL in order to access schema files over https.

User Interface and Visualization Drivers

In addition to the packages listed below for individual drivers, we strongly recommend installing the drivers for the video card on your system (e.g. NVIDIA).

Users of the Wayland window manager will need XWayland for the X11-based drivers below. Our experience is that this solution does co-work with Qt.

- Qt5 User Interface and Visualization (*All Platforms*)
  - Qt5 headers and libraries
    - You will need to register personally as an open software developer to obtain a free personal version of Qt.
    - For most platforms you can get a binary installation package.
    - If you are installing individual Qt5 modules, the set required by Geant4 is `Qt5Core`, `Qt5Gui` `Qt5Widgets`, `Qt5OpenGL`, and `Qt5PrintSupport`. Use of the optional and experimental Qt3D driver additionally requires Qt5 5.15 and the `Qt53DCore`, `Qt53DExtras` and `Qt53DRender` modules.

        * If you need to compile, Qt5 should preferably be compiled against the same C++ Standard as Geant4 (C++11 by default), but this is not required as its ABI is binary compatible between standards.
    - OpenGL or MesaGL headers and libraries.
- X11 OpenGL Visualization (*Linux and macOS*)
    - X11 headers and libraries (XQuartz on macOS).
    - OpenGL or MesaGL headers and libraries.
- WIN32 OpenGL Visualization (*Windows*)
    - OpenGL or MesaGL headers and libraries.
    - Visual Studio supplies a basic install of OpenGL.
- X11 RayTracer Visualization (*Linux and macOS*)
    - X11 headers and libraries (XQuartz on macOS).
- Open Inventor Visualization (*All Platforms*)
    - Coin3D libraries and headers, version 4.0.0 or newer, plus one the bindings:
    - SoXt libraries and headers, version 1.4.0 or newer
    - SoQt libraries and headers, version 1.6.0 or newer
    - SoWin libraries and headers, version 1.4.0 or newer (*SoWin support is provisional*)
    - See also *Installing Coin3D and SoXt/Qt Bindings*
- Motif User Interface and Visualization (*Linux and macOS*)
    - Motif headers and libraries.
    - X11 headers and libraries (XQuartz on macOS).
    - OpenGL or MesaGL headers and libraries.

Analysis Features and Backends

The Geant4 analysis library provides a lightweight interface for storing quantities and plots with various backends for persistency (e.g. plain text, XML). Whilst the choice of backend and linking is deferred to the user as required for their application, the following features require presence of additional software when compiling Geant4:

- Freetype Font Rendering Support (*Linux and macOS*)
    - Freetype headers and libraries.

Python Bindings

The Python bindings to Geant4, Geant4Py, may now be built as part of the main Geant4 build and require:

- Python 3.X interpreter, library, and headers
- Boost Python 1.69 library and headers, compiled/linked against the required Python version.

Advanced/Experimental Features

Warning: These features are for advanced users only. Note that HDF5 use is experimental.

- VecGeom Replacements for Geant4 solids
  - VecGeom headers and libraries, version 1.1.8 or newer, compiled against the same C++ Standard as Geant4 (C++11 by default)
- TiMemory profiling for Geant4 kernel and applications
  - TiMemory headers and library, compiled against the same C++ Standard as Geant4 (C++11 by default)
- HDF5 Persistency for Geant4 Analysis module
  - HDF5 1.8 or higher C headers and libraries.
  - If Geant4 is built with multithreading support, then the used HDF5 install *must* have been compiled with thread safety enabled.

Software Suggested for Use With Geant4 Applications

Geant4 includes many cross-platform file-based visualization drivers, together with the lightweight inexlib library for basic analysis. Geant4 does not require any additional software over and above that listed in *Geant4 System/Software Prerequisites* to *build and install* these components.

However, you may wish to install the third-party software suggested below to make use of these components when running your Geant4 application. We again emphasise that you do not need these packages to build and install Geant4. Also note that Geant4 cannot provide support on installing or using these packages. Any issues here should be reported to the developers of the package.

- DAWN postscript renderer (for use with DAWN visualization driver).
- HepRApp Browser (for use with HepRepFile visualization driver).
- WIRED4 JAS Plug-In (for use with HepRepXML visualization driver).
- VRML Browser (for use with VRML visualization driver).
- OpenScientist interactive environment for analysis.
- AIDA implementation such as OpenScientist, JAS3 or rAIDA.
- gMocren volume visualizer for Geant4 medical simulations.

For more details on Geant4's visualization and analysis components, you should consult the relevant sections in the Geant4 User's Guide for Application Developers.

2.6.2 Building and Installing from Source

On Unix Platforms

Unpack the Geant4 source package `geant4.10.07.tar.gz` to a location of your choice. For illustration *only*, this guide will assume it's been unpacked in a directory named `/path/to`, so that the Geant4 source package sits in a subdirectory

```
/path/to/geant4.10.07
```

We refer to this directory as the *source directory*. The next step is to create a directory in which to configure and run the build and store the build products. This directory should not be the same as, or inside, the source directory. In this guide, we create this *build directory* alongside our source directory:

```
$ cd /path/to
$ mkdir geant4.10.07-build
$ ls
geant4.10.07  geant4.10.07-build
```

To configure the build, change into the build directory and run CMake:

```
$ cd /path/to/geant4.10.07-build
$ cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.07-install /path/to/geant4.
↪10.07
```

Here, the CMake Variable `CMAKE_INSTALL_PREFIX` is used to set the *install directory*, the directory under which the Geant4 libraries, headers and support files will be installed. It must be supplied as an absolute path. The second argument to CMake is the path to the source directory. In this example, we have used the absolute path to the source directory, but you can also use the relative path from your build directory to your source directory.

Additional arguments may be passed to CMake to activate optional components of Geant4, such as visualization drivers, or tune the build and install parameters. See *Geant4 Build Options* for details of these options. If you run CMake and decide afterwards you want to activate additional options, simply rerun CMake in the build directory, passing it the extra options plus the build directory path. For example, after running CMake as above, you may wish to activate the installation of Geant4's datasets, so you would run (in the build directory)

```
$ cd /path/to/geant4.10.07-build
$ cmake -DGEANT4_INSTALL_DATA=ON .
```

On executing the CMake command, it will run to configure the build and generate Unix Makefiles to perform the actual build. CMake has the capability to generate buildscripts for other tools, such as Eclipse and Xcode, but please note that *we do not support user installs of Geant4 with these tools*. On Linux, you will see output similar to:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.07-install /path/to/geant4.
↪10.07
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is GNU 4.9.2
-- Check for working C compiler: /usr/bin/gcc-4.9
-- Check for working C compiler: /usr/bin/gcc-4.9 -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/g++-4.9
-- Check for working CXX compiler: /usr/bin/g++-4.9 -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
```

```
-- Found EXPAT: /usr/lib64/libexpat.so (found version "2.0.1")
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for stddef.h
-- Looking for stddef.h - found
-- Check size of off64_t
-- Check size of off64_t - done
-- Looking for fseeko
-- Looking for fseeko - found
-- Looking for unistd.h
-- Looking for unistd.h - found
-- Pre-configuring dataset G4NDL (4.6)
-- Pre-configuring dataset G4EMLOW (7.13)
-- Pre-configuring dataset PhotonEvaporation (5.7)
-- Pre-configuring dataset RadioactiveDecay (5.6)
-- Pre-configuring dataset G4PARTICLEXS (3.1)
-- Pre-configuring dataset G4PII (1.3)
-- Pre-configuring dataset RealSurface (2.2)
-- Pre-configuring dataset G4SAIDDATA (2.0))
-- Pre-configuring dataset G4ABLA (3.1)
-- Pre-configuring dataset G4INCL (1.0)
-- Pre-configuring dataset G4ENSDFSTATE (2.3)
  *WARNING*
    Geant4 has been pre-configured to look for datasets
    in the directory:

    /path/to/geant4.10.07-install/share/Geant4-10.7.0/data

    but the following datasets are NOT present on disk at
    that location:

    G4NDL (4.6)
    G4EMLOW (7.13)
    PhotonEvaporation (5.7)
    RadioactiveDecay (5.6)
    G4PARTICLEXS (3.1)
    G4PII (1.3)
    RealSurface (2.2)
    G4SAIDDATA (2.0))
    G4ABLA (3.1)
    G4INCL (1.0)
    G4ENSDFSTATE (2.3)

    If you want to have these datasets installed automatically
    simply re-run cmake and set the GEANT4_INSTALL_DATA
    variable to ON. This will configure the build to download
    and install these datasets for you. For example, on the
    command line, do:

    cmake -DGEANT4_INSTALL_DATA=ON <otherargs>

    The variable can also be toggled in ccmake or cmake-gui.
```

```
    If you're running on a Windows system, this is the best
    solution as CMake will unpack the datasets for you
    without any further software being required

    Alternatively, you can install these datasets manually
    now or after you have installed Geant4. To do this,
    download the following files:

    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4NDL.4.6.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4EMLOW.7.13.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PhotonEvaporation.5.
 ↪7.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4RadioactiveDecay.
 ↪5.6.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PARTICLEXS.3.1.tar.
 ↪gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PII.1.3.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4RealSurface.2.2.
 ↪tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4SAIDDATA.2.0.tar.
 ↪gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4ABLA.3.1.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4ENSDFSTATE.2.3.tar.
 ↪gz

    and unpack them under the directory:

    /path/to/geant4.10.07-install/share/Geant4-10.7.0/data

    As we supply the datasets packed in gzipped tar files,
    you will need the 'tar' utility to unpack them.

    Nota bene: Missing datasets will not affect or break
    compilation and installation of the Geant4
    libraries.

-- The following Geant4 features are enabled:
GEANT4_USE_SYSTEM_EXPAT: Using system EXPAT library

-- Configuring done
-- Generating done
-- Build files have been written to: /path/to/geant4.10.07-build
```

The exact output will differ depending on the platform/compiler in use, but the last three lines should be the same to

within path differences. These indicate a successful configuration.

The warning message about datasets is simply an advisory. Due to the size of the datasets, Geant4 will try and reuse any datasets it can find under the data installation prefix, in our example case `/path/to/geant4.10.07-install/share/Geant4-10.7.0/data`. If any datasets are not found here, it will pre-configure the setup scripts for using Geant4 (described in *Postinstall Setup*) to point to this location and emit the message to advise you on the steps you need to take to manually install the datasets at a time of your convenience.

Datasets are *not* required to be present to build Geant4, but may be required to run your application, depending on the physics models you use. If you wish to download and install the datasets automatically as part of your build of Geant4, simply add the option `-DGEANT4_INSTALL_DATA=ON` to the arguments passed to CMake. Note that this requires a working network connection and will download around 0.5GB of data. If you already have the datasets present on your system, you can point Geant4 to their location. See the `GEANT4_INSTALL_DATADIR` option described *Standard Options* for more details.

If you see any errors at this point, carefully check the error messages output by CMake, and check your install of CMake and C++ compiler first. The default configuration of Geant4 is very simple, and provided CMake and the compiler are installed correctly, you should not see errors.

After the configuration has run, CMake will have generated Unix Makefiles for building Geant4. To run the build, simply execute make in the build directory:

```
$ make -jN
```

where `N` is the number of parallel jobs you require (e.g. if your machine has a dual core processor, you could set `N` to 2).

The build will now run, and will output information on the progress of the build and current operations. If you need more output to help resolve issues or simply for information, run make as:

```
$ make -jN VERBOSE=1
```

Once the build has completed, you can install Geant4 to the directory you specified earlier in `CMAKE_INSTALL_PREFIX` by running:

```
$ make install
```

in the build directory. The libraries, headers and resource files are installed under your chosen install prefix in a standard Unix-style hierarchy of directories, described below in *Postinstall Setup*. If you are performing a staged install for packaging or deployment, the CMake generated Makefiles support the `DESTDIR` variable for copying to a temporary location. To uninstall Geant4 you can run:

```
$ make uninstall
```

which will remove all installed files but not any installed directories.

On Windows Platforms

The easiest way to build and install Geant4 from source on Windows platforms is to use the Windows command line program `cmd` plus CMake's command line interface to the MSBuild tool supplied with Visual Studio. Whilst the full Visual Studio GUI can be used, `cmd` and CMake/MSBuild provide a simpler interface and the commands can be used inside scripts. You can also use PowerShell instead of `cmd` if you prefer, and the instructions below should transfer barring minor syntax differences. Builds of Geant4 using Cygwin or MinGW with their own compilers or the Microsoft C++ Compiler are neither supported or tested, though the CMake system is expected to work under these toolchains. If you are using these tools via their native shells and with their own versions of CMake, then the instructions for building and installing on Unix platforms *On Unix Platforms* can be used.

To ensure that the appropriate Visual Studio paths and settings are set up for building, open a Visual Studio Developer `cmd` window from *Start → Visual Studio 201X → Visual Studio Tools → Developer Command Prompt for VS201X*. After running this, confirm that you have the MSVC compiler available by running the `cl` command, and you should see (NB, in the following, the `cmd` prompt is shown as a > for clarity):

```
> cl
Microsoft (R) C/C++ Optimizing Compiler Version 19.11.25547 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

usage: cl [ option... ] filename... [ /link linkoption... ]
```

The exact version number of `cl` may differ slightly, but for Visual Studio 2019 the first element of the Compiler Version should be at least `19`.

To begin building Geant4, unpack the source package `geant4_10_07.zip` to a location of your choice. For illustration *only*, this guide will assume it's been unpacked in a directory named `C:\Users\YourUsername\Geant4`, so that the Geant4 source package sits in a subdirectory `C:\Users\YourUsername\Geant4\geant4_10_07`

We refer to this directory as the *source directory*. The next step is to create a directory in which to configure and run the build and store the build products. This directory should not be the same as, or inside, the source directory. In this guide, we create this *build directory* alongside our source directory:

```
> cd %HOMEPATH%\Geant4
> dir /B
geant4_10_07

> mkdir geant4_10_07-build
> dir /B
geant4_10_07
geant4_10_07-build
```

To configure the build, change into the build directory and run CMake:

```
> cd %HOMEPATH%\Geant4\geant4_10_07-build
> cmake -DCMAKE_INSTALL_PREFIX="%HOMEPATH%\Geant4\geant4_10_07-install" "
↪%HOMEPATH%\Geant4\geant4_10_07"
```

Here, the CMake Variable `CMAKE_INSTALL_PREFIX` is used to set the *install directory*, the directory under which the Geant4 libraries, headers and support files will be installed. It must be supplied as an absolute path. The second argument to CMake is the path to the source directory. In this example, we have used the absolute path to the source directory, but you can also use the relative path from your build directory to your source directory. Paths should be quoted in case they contain spaces.

Additional arguments may be passed to CMake to activate optional components of Geant4, such as visualization drivers, or tune the build and install parameters. See *Geant4 Build Options* for details of these options. If you run CMake and decide afterwards you want to activate additional options, simply rerun CMake in the build directory, passing it the extra options plus the build directory. For example, after running CMake as above, you may wish to activate the installation of Geant4's datasets, so you would run

```
> cd %HOMEPATH%\Geant4\geant4_10_07-build
> cmake -DGEANT4_INSTALL_DATA=ON .
```

On executing the CMake command, it will run to configure the build and generate Visual Studio Project files to perform the actual build. CMake has the capability to generate buildscripts for other tools, such as NMake and Ninja, but please note that *these are not supported for builds of Geant4 on Windows yet*. With Visual Studio, you should see output similar to

```
> cmake -DCMAKE_INSTALL_PREFIX="%HOMEPATH%\Geant4\geant4_10_07-install" "
↪%HOMEPATH%\Geant4\geant4_10_07"
-- Building for: Visual Studio 15 2017
```

```
-- The C compiler identification is MSVC 19.11.25547.0
-- The CXX compiler identification is MSVC 19.11.25547.0
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual␣
↪Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x86/cl.exe
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual␣
↪Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x86/cl.exe --␣
↪works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual␣
↪Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x86/cl.exe
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual␣
↪Studio/2017/Community/VC/Tools/MSVC/14.11.25503/bin/Hostx86/x86/cl.exe --␣
↪works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for dlfcn.h
-- Looking for dlfcn.h - not found
-- Looking for fcntl.h
-- Looking for fcntl.h - found
-- Looking for inttypes.h
-- Looking for inttypes.h - found
-- Looking for memory.h
-- Looking for memory.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for stdlib.h
-- Looking for stdlib.h - found
-- Looking for strings.h
-- Looking for strings.h - not found
-- Looking for string.h
-- Looking for string.h - found
-- Looking for sys/stat.h
-- Looking for sys/stat.h - found
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for unistd.h
-- Looking for unistd.h - not found
-- Looking for getpagesize
-- Looking for getpagesize - not found
-- Looking for bcopy
-- Looking for bcopy - not found
-- Looking for memmove
-- Looking for memmove - found
-- Looking for mmap
-- Looking for mmap - not found
-- Looking for 4 include files stdlib.h, ..., float.h
-- Looking for 4 include files stdlib.h, ..., float.h - found
-- Check if the system is big endian
-- Searching 16 bit integer
-- Looking for stddef.h
-- Looking for stddef.h - found
```

```
-- Check size of unsigned short
-- Check size of unsigned short - done
-- Using unsigned short
-- Check if the system is big endian - little endian
-- Looking for off_t
-- Looking for off_t - not found
-- Looking for size_t
-- Looking for size_t - not found
-- Check size of off64_t
-- Check size of off64_t - failed
-- Looking for fseeko
-- Looking for fseeko - not found
-- Looking for unistd.h
-- Looking for unistd.h - not found
-- Pre-configuring dataset G4NDL (4.6)
-- Pre-configuring dataset G4EMLOW (7.13)
-- Pre-configuring dataset PhotonEvaporation (5.7)
-- Pre-configuring dataset RadioactiveDecay (5.6)
-- Pre-configuring dataset G4PARTICLEXS (3.1)
-- Pre-configuring dataset G4PII (1.3)
-- Pre-configuring dataset RealSurface (2.2)
-- Pre-configuring dataset G4SAIDDATA (2.0))
-- Pre-configuring dataset G4ABLA (3.1)
-- Pre-configuring dataset G4INCL (1.0)
-- Pre-configuring dataset G4ENSDFSTATE (2.3)
  *WARNING*
    Geant4 has been pre-configured to look for datasets
    in the directory:

    /Users/YourUsername/Geant4/geant4_10_07-install/share/Geant4-10.7.0/data

    but the following datasets are NOT present on disk at
    that location:

    G4NDL (4.6)
    G4EMLOW (7.13)
    PhotonEvaporation (5.7)
    RadioactiveDecay (5.6)
    G4PARTICLEXS (3.1)
    G4PII (1.3)
    RealSurface (2.2)
    G4SAIDDATA (2.0))
    G4ABLA (3.1)
    G4INCL (1.0)
    G4ENSDFSTATE (2.3)

    If you want to have these datasets installed automatically
    simply re-run cmake and set the GEANT4_INSTALL_DATA
    variable to ON. This will configure the build to download
    and install these datasets for you. For example, on the
    command line, do:

    cmake -DGEANT4_INSTALL_DATA=ON <otherargs>
```

```
    The variable can also be toggled in ccmake or cmake-gui.
    If you're running on a Windows system, this is the best
    solution as CMake will unpack the datasets for you
    without any further software being required

    Alternatively, you can install these datasets manually
    now or after you have installed Geant4. To do this,
    download the following files:

    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4NDL.4.6.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4EMLOW.7.13.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PhotonEvaporation.5.
 ↪7.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4RadioactiveDecay.
 ↪5.6.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PARTICLEXS.3.1.tar.
 ↪gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4PII.1.3.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4RealSurface.2.2.
 ↪tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4SAIDDATA.2.0.tar.
 ↪gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4ABLA.3.1.tar.gz
    https://cern.ch/geant4\unhbox\voidb@x\kern\z@\char`\protect\
discretionary{\char\defaulthyphenchar}{}{}data/datasets/G4ENSDFSTATE.2.3.tar.
 ↪gz

    and unpack them under the directory:

    /Users/YourUsername/Geant4/geant4_10_07-install/share/Geant4-10.7.0/data

    As we supply the datasets packed in gzipped tar files,
    you will need the 'tar' utility to unpack them.

    Nota bene: Missing datasets will not affect or break
    compilation and installation of the Geant4
    libraries.

-- The following Geant4 features are enabled:

-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/YourUsername/Geant4/geant4_10_
 ↪04-build
```

The output will differ slightly for Visual Studio 2019, and due to the source/build paths being for illustration only, but the last three lines at least should be the same to within path differences. These indicate a successful configuration.

The warning message about datasets is simply an advisory. Due to the size of the datasets, Geant4 will try and reuse any datasets it can find under the data installation prefix, in our example case `C:\Users\YourUsername\Geant4\geant4_10_07-install\share\Geant4-10.7.0\data`. If any datasets are not found here, the message is emitted to advise you of the steps you need to take to manually install the datasets at a time of your convenience.

Datasets are *not* required to be present to build Geant4, but may be required to run your application, depending on the physics models you use. If you wish to download and install the datasets automatically as part of your build of Geant4, simply add the option `-DGEANT4_INSTALL_DATA=ON` to the arguments passed to CMake. Note that this requires a working network connection and will download around 0.5GB of data. If you already have the datasets present on your system, you can point Geant4 to their location. See the `GEANT4_INSTALL_DATADIR` option described *Standard Options* for more details.

If you see any errors at this point, carefully check the error messages output by CMake, and check your install of CMake and Visual Studio first. The default configuration of Geant4 is very simple, and provided CMake and Visual Studio are installed correctly, you should not see errors.

After the configuration has run, CMake will have generated Visual Studio Solution files for building Geant4. CMake itself can be used to run the build by executing the command:

```
> cmake --build . --config Release
```

Here, the `--build` argument takes the path to the build directory, in this case we are running from the build directory so it is just the current working directory. The `--config` argument takes the configuration we want to build (Visual Studio, unlike Make, can support multiple configurations in the same project) and `Release` is chosen to provide fully optimised libraries for best performance. If you are developing applications and require debugging information, then you should change this argument to `RelWithDebInfo`.

The build will now run, and will output information on the progress of the build and current operations. By default, Visual Studio Solutions do not enable parallel compilation of files for faster builds. Geant4's CMake system provides an option to enable this, so if you have a multicore system, you can add the option `-DGEANT4_BUILD_MSVC_MP=ON` to the arguments passed to CMake. Once the build has completed the headers, libraries and support files can be installed by running the command:

```
> cmake --build . --config Release --target install
```

This command may also be invoked immediately after configuration to build and install Geant4 in one step. The file and directory structure of the installation follows that of the Unix build, and is described in *Postinstall Setup*.

Geant4 Build Options

Both *On Unix Platforms* and *On Windows Platforms* give the minimal procedure to build and install Geant4 on these platforms. Many additional options can be passed to CMake to adjust the way Geant4 is built and installed and to enable optional components.

Options are divided into *Standard Options*, which any user or developer can set directly, and *Advanced Options*, which in general are only needed by advanced users, developers or to give very fine control over the build and install. Some options enable components of Geant4 which require external software (as listed in *Geant4 System/Software Prerequisites*). If these options are enabled, the required software will be searched for, and hence there are also options which control where CMake should look for these packages. If a required software package is not found, then CMake will exit with an error message detailing what was not found.

These options may be set by passing their name and value to the `cmake` command via `-D` flags. For example:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/opt/geant4 -DGEANT4_USE_GDML=ON /path/to/geant4-source
```

would configure the build of Geant4 for installation under `/opt/geant4` and compilation of support for GDML.

If you have already created a build directory and used CMake to configure the build, you can always rerun CMake in that directory with new options to regenerate the buildscripts (Makefiles or IDE solutions). You can also *deactivate* a previously selected option to remove a component from the build. For example, if we had configured a build to enable GDML support and wanted to remove it, we could run:

```
$ cmake -DGEANT4_USE_GDML=OFF .
```

Note that this assumes we are running `cmake` in a previously configured build directory so we only need pass the current working directory rather than the full source directory path.

If you reconfigure to *unset* an option and rebuild and reinstall, your install may contain files installed by the previously set option (for example headers). In this case, you should build the `uninstall` target before reconfiguring to guarantee removal of these files.

CMake also provides Curses (UNIX only) and Qt (UNIX and Windows) based Terminal/GUI interfaces which may be used to browse and set options. Please see the CMake documentation for more information on these interfaces.

Standard Options

We list standard options here in logical order. If you use CMake's curses or GUI interfaces, they will be listed alphabetically. Where third-party software requirements are listed, please consult *Geant4 System/Software Prerequisites* for links and information on these packages.

- `CMAKE_INSTALL_PREFIX`
  - Sets the installation prefix for Geant4. Equivalent to `--prefix` in Autotools. The default is platform dependent:

    Unix: `/usr/local`

    Windows: `C:\Program Files\Geant4`

    It should be supplied as an absolute path, otherwise CMake will interpret its value relative to your build directory.

    See also the `CMAKE_INSTALL_XXXDIR` *Advanced Options* for fine control of installation locations.
- `CMAKE_BUILD_TYPE : (DEFAULT : Release)`
  - Sets the type of build. It defaults to `Release` which gives a fully optimised build without debugging symbols. The most useful values are:

    `Release` : Optimised build, no debugging symbols

    `Debug` : Debugging symbols, no optimisation

    `RelWithDebInfo` : Optimised build with debugging symbols

    See *Options for Changing the Compiler and Build Flags* for the compiler flags used in each mode.

    Note that if you use a build system which supports multiconfiguration builds (e.g. Xcode and Visual Studio), this variable has no effect. For these systems, all build types are available inside the CMake generated project and can be selected in the tool itself or via the `--config` argument to `cmake --build`, e.g. `cmake --build . --config Debug`.
- `GEANT4_BUILD_MULTITHREADED : (DEFAULT : OFF)`

  - If set to ON, build Geant4 libraries with support for multithreading. At present, this is only supported on Unix systems.

    Requires: Compiler/C++ Standard Library with support for C++ Threading Support

- GEANT4_INSTALL_DATA : (DEFAULT : OFF)
  - If set to ON, download and install any Geant4 datasets missing from GEANT4_INSTALL_DATADIR. Each dataset will be unpacked and installed in the directory pointed to by GEANT4_INSTALL_DATADIR.

    Requires: A working network connection. It is highly recommended to switch this option on if you have a network connection to give the best integration with application development.

- GEANT4_INSTALL_DATADIR : (DEFAULT : CMAKE_INSTALL_DATAROOTDIR)
  - Installation directory for Geant4 datasets. It can be supplied as a path relative to CMAKE_INSTALL_PREFIX or as an absolute path. It is always searched for existing datasets, which if present will be reused.

- GEANT4_USE_G3TOG4 : (DEFAULT : OFF)
  - If set to ON, build the G3ToG4 library for reading ASCII call list files generated from Geant3 geometries.

- GEANT4_USE_GDML : (DEFAULT : OFF)
  - If set to ON, build the G4persistency library with support for GDML.

    Requires: Xerces-C++ libraries and headers. See the CMAKE_PREFIX_PATH *Advanced Options* or the CMake FindXercesC documentation if CMake has trouble locating Xerces-C

- GEANT4_USE_INVENTOR (DEFAULT : OFF)
  - If set to ON, build the X11/Win32 OpenInventor visualization driver.

    Requires: Coin3D Open Inventor implementation, SoXt (Unix) or SoWin (Windows) binding, and OpenGL libraries and headers.

    See the CMAKE_PREFIX_PATH *Advanced Options* if CMake has trouble locating Coin3D or SoXt/SoWin.

    Known Issue: Use of clang compiler and Debug build mode will cause the Inventor driver build to fail with errors relating to Inventor specific debugging functions.

- GEANT4_USE_INVENTOR_QT (DEFAULT : OFF)
  - If set to ON, build the OpenInventorQt visualization driver.

    Note that enabling this option will disable build of the X11/Win OpenInventor driver as only one binding can be present.

    Requires: Coin3D Open Inventor implementation plus SoQt binding and Qt5 libraries and headers.

    See the CMAKE_PREFIX_PATH *Advanced Options* if CMake has trouble locating Coin3D or SoQt.

- GEANT4_USE_OPENGL_WIN32 (DEFAULT : OFF, Windows Only)
  - If set to ON, build the Win32 OpenGL visualization driver.

    Requires: OpenGL libraries and headers.

- GEANT4_USE_OPENGL_X11 (DEFAULT : OFF, Unix Only)
  - If set to ON, build the X11 OpenGL visualization driver.

    Requires: X11 and OpenGL libraries and headers.

- GEANT4_USE_PYTHON (DEFAULT : OFF)

  - If set to ON, build the Geant4Py Python binds to Geant4.

    Requires: Python 3 and Boost Python libraries and headers. If GEANT4_BUILD_MULTITHREADED is ON, then GEANT4_BUILD_TLS_MODEL must be set to global-dynamic to allow runtime loading of the modules.

    See the CMAKE_PREFIX_PATH *Advanced Options* or the CMake documentation for FindBoost , FindPythonInterp , and FindPythonLibs if CMake has trouble locating Boost or Python3.

- GEANT4_USE_QT (DEFAULT : OFF)
  - If set to ON, build Qt5 User Interface and Visualization drivers.

    Requires: Qt5 and OpenGL libraries and headers.

    See the CMAKE_PREFIX_PATH variable under *Advanced Options* if CMake has trouble locating Qt.

- GEANT4_USE_RAYTRACER_X11 (DEFAULT : OFF, Unix only)
  - If set to ON, build RayTracer visualization driver with X11 support.

    Requires: X11 libraries and headers.

- GEANT4_USE_SYSTEM_CLHEP (DEFAULT : OFF)
  - If set to ON, build Geant4 with an external install of CLHEP. You *should not* set this unless your usage of Geant4 mandates a specific external CLHEP installation (e.g. if your project's software uses CLHEP in other tools and requires consistent use of the same CLHEP across the software stack).

    Requires: CLHEP libraries and headers. See the CMAKE_PREFIX_PATH *Advanced Options* if CMake has trouble locating CLHEP.

- GEANT4_USE_SYSTEM_EXPAT (DEFAULT : ON)
  - If set to ON, build Geant4 with an external install of Expat. Whilst Expat is installed on the vast majority of systems, it may be missing in certain instances. In these cases, simply switch this option to OFF and Geant4 will build and use its internal version of Expat.

    Requires: Expat library and headers. See the CMAKE_PREFIX_PATH *Advanced Options* or the CMake documentation for FindEXPAT if CMake has trouble locating Expat.

- GEANT4_USE_SYSTEM_ZLIB (DEFAULT : OFF)
  - If set to ON, build Geant4 with an external install of zlib.

    Requires: Zlib library and headers. See the CMAKE_PREFIX_PATH *Advanced Options* or the CMake documentation for FindEXPAT if CMake has trouble locating ZLIB.

- GEANT4_USE_TBB (DEFAULT : ON)
  - If set to ON, use Intel TBB as the Tasking backend for PTL.

    Requires: Intel TBB libraries and headers. See the CMAKE_PREFIX_PATH *Advanced Options* if CMake has trouble locating TBB

- GEANT4_USE_XM (DEFAULT : OFF, Unix Only)
  - If set to ON, build Motif User Interface and Visualization drivers.

    Requires: Motif and OpenGL libraries and headers. See the CMAKE_PREFIX_PATH *Advanced Options* or the CMake documentation for FindMotif if CMake has trouble locating Motif.

Advanced Options

Most installs should never need to touch these options, and are primarily to give advanced users more control over the build, enable experimental features, and to help CMake locate needed software packages. Advanced options and variables can be set like the standard ones listed earlier using `-D` arguments to `cmake`. In CMake's curses and GUI interfaces these options can be displayed by pressing `t` in `ccmake`, or clicking the 'advanced' check box in the CMake GUI.

In the list below, we only list those options most relevant for Geant4. Many additional core CMake variables are available, for which you should consult the Reference Documentation section of the main CMake documentation , and specifically the sections on Variables. The following list is presented in semi-alphabetical order, with grouping by task where appropriate.

- `BUILD_SHARED_LIBS : (DEFAULT : ON)`
  - If set to `ON` build Geant4 shared libraries.
- `BUILD_STATIC_LIBS : (DEFAULT : OFF)`
  - If set to `ON`, build Geant4 static libraries.
- `CMAKE_INSTALL_BINDIR : (DEFAULT : bin)`
  - Installation directory for Geant4 Toolkit executables. It can be supplied as a path relative to `CMAKE_INSTALL_PREFIX` or as an absolute path.
- `CMAKE_INSTALL_INCLUDEDIR : (DEFAULT : include)`
  - Installation directory for Geant4 C/C++ headers. It can be supplied as a path relative to `CMAKE_INSTALL_PREFIX` or as an absolute path. The headers will always be installed in a subdirectory of `CMAKE_INSTALL_INCLUDEDIR` named `Geant4`.
- `CMAKE_INSTALL_LIBDIR : (DEFAULT : lib(+?SUFFIX))`
  - Installation directory for object code libraries. It can be supplied as a path relative to `CMAKE_INSTALL_PREFIX`, or an absolute path. When the default is used, `SUFFIX` will be set to `64` on 64bit Linux platforms apart from Debian systems.
- `CMAKE_INSTALL_PYTHONDIR : (DEFAULT : CMAKE_INSTALL_LIBDIR/python3.<PYMINOR>/site-packages)`
  - Installation directory for the Geant4 python package/bindings. It can be supplied as a path relative to `CMAKE_INSTALL_PREFIX`, or an absolute path.
- `CMAKE_INSTALL_DATAROOTDIR : (DEFAULT : share)`
  - Installation directory for read-only architecture-independent data files. It can be supplied as a path relative to `CMAKE_INSTALL_PREFIX`, or an absolute path.
- `GEANT4_INSTALL_DATA_TIMEOUT : (DEFAULT : 1500)`
  - Sets the time in seconds allowed for download of each Geant4 dataset. The value can be increased if you are on a slow network connection and require more time to download.
- `GEANT4_INSTALL_EXAMPLES : (DEFAULT : ON)`
  - Set to `OFF` to prevent installation of the source code and documentation for the Geant4 examples.
- `GEANT4_BUILD_CXXSTD : (DEFAULT : 11 (UNIX), 17 (Windows))`
  - Compile Geant4 against given C++ standard (`11`, `14`, or `17`). Geant4 is written in C++11, and you should use this option if your application requires support for the newer standard. If you set the variable to a standard the compiler does not support, an error will be emitted.

    Requires: C++ Compiler with support for the requested standard.

- GEANT4_BUILD_MSVC_MP : (Windows Only, DEFAULT : OFF)
  - If set to ON, add /MP to CMAKE_CXX_FLAGS to enable file level parallel compilation when using MSVC and MSBuild. Note that this only works when building Geant4 using Visual Studio Solutions.
- GEANT4_BUILD_TLS_MODEL : (DEFAULT : initial-exec)
  - If building Geant4 with multithreading support, use a specific model for Thread Local Storage (initial-exec, local-exec, global-dynamic, local-dynamic or auto). If you set the variable to a model unknown to the compiler, an error will be emitted.

    Geant4's default model of initial-exec is chosen to give the best performance under a wide variety of use cases.

    The special auto value leaves the choice of TLS model to the compiler.

    Requires: GEANT4_BUILD_MULTITHREADED set to ON
- GEANT4_BUILD_STORE_TRAJECTORY : (DEFAULT : ON)
  - If set to ON, store trajectories in event processing. It can be switched to OFF to give a degree of performance improvement, but you will *not* be able to visualise events.
- GEANT4_BUILD_VERBOSE_CODE : (DEFAULT : ON)
  - If set to ON, build Geant4 libraries with extra verbosity. It can be switched to OFF to give a degree of performance improvement, but you will not have as much information output should you run into problems or need to debug.
- GEANT4_BUILD_BUILTIN_BACKTRACE : (DEFAULT : OFF)
  - If set to ON, build Geant4 Run Manager with signal handling using G4Backtrace. This provides simple backtrace reporting, but should not be set to ON if you application implements/requires its own signal handling.
- GEANT4_BUILD_PHP_AS_HP : (DEFAULT : OFF)
  - If set to ON, build the ParticleHP model as HP.
- GEANT4_USE_SMARTSTACK : (DEFAULT : OFF)
  - If set to ON, use G4SmartStack in G4TrackingManager.
- GEANT4_USE_SYSTEM_PTL (DEFAULT : OFF)
  - If set to ON, build Geant4 with an external install of PTL.

    Requires: PTL library and headers. See the CMAKE_PREFIX_PATH *Advanced Options* if CMake has trouble locating PTL.
- GEANT4_ENABLE_TESTING : (DEFAULT : OFF)
  - If set to ON, build and run Geant4 testing suites.

    WARNING: this option is for Geant4 system testing only and is not intended for use by users. No support is, or will be, provided for user builds with this option.
- GEANT4_USE_NETWORKDAWN : (DEFAULT : OFF, Unix Only)
  - If set to ON, build network server/client support for DAWN visualization driver. You do **not** need this to view DAWN files.
- GEANT4_USE_NETWORKVRML : (DEFAULT : OFF, Unix Only)
  - If set to ON, build network server/client support for VRML visualization driver. You do **not** need this to view VRML files.

- GEANT4_USE_FREETYPE : (DEFAULT : OFF)
  - If set to ON, build Geant4 Analysis library with support for Freetype font rendering.

    Requires: Freetype libraries and headers.
- GEANT4_USE_HDF5 : (DEFAULT : OFF)
  - If set to ON, build Geant4 Analysis library with support for HDF5 persistency.

    WARNING: use of HDF5 is experimental and should be used with caution.

    Requires: HDF5 C libraries and headers, compiled in threadsafe mode if Geant4 is built with multithreading support.
- GEANT4_USE_USOLIDS : (DEFAULT : OFF)
  - If set to ON, replace Geant4 solids with VecGeom equivalents.

    WARNING: the use of VecGeom is experimental and should be used with caution.

    Requires: VecGeom libraries and headers.
- GEANT4_USE_TIMEMORY : (DEFAULT : OFF)
  - If set to ON, build Geant4 with support for memory/CPU profiling with TiMemory.

    Requires: TiMemory library and headers.
- GEANT4_INSTALL_PACKAGE_CACHE : (DEFAULT : ON)
  - If set to ON, install a file containing the locations of external dependencies as used when building this install of Geant4.

    Geant4 10.6 and newer use CMake Imported Targets to link to external dependencies such as CLHEP, avoiding direct hard-coding of paths into the Geant4 install. The package cache file is installed by default to retain the pre-10.6 behaviour for basic installs, and stores these paths for reuse.

    If you are packaging (e.g. rpm, deb, spack, Homebrew, etc) Geant4, you can set this option to OFF to disable install of the cache file and make the package *relocatable*. Your install of Geant4 will then rely on CMAKE_PREFIX_PATH and other variables used by CMake to re-find any needed dependencies. Alternately, if your packaging system allows install-time patching, you may patch the cache file at that time with known paths to the dependencies at that point. In both cases, it is *your* responsibility to setup/patch the environment/cache file correctly and as appropriate.

    WARNING: This option is only intended for the packaging case outlined above, and is not recommended for single-user installs of Geant4.*
- GEANT4_INSTALL_DATASETS_TENDL : (DEFAULT :OFF)
  - If set to ON, enable download and install of the optional TENDL dataset for use with the
- CMAKE_PREFIX_PATH
  - If you have software packages required by Geant4 installed in custom locations, this variable can be set to list these prefixes to help CMake locate the packages. For example, if Xerces-C is needed and installed in /custom/xerces-c, then CMAKE_PREFIX_PATH could be set on the command line as:

    ```
    cmake -DCMAKE_PREFIX_PATH="/custom/xerces-c" <otherargs>
    ```

    Additional paths may be added, separated by semicolons, e.g.:

    ```
    cmake -DCMAKE_PREFIX_PATH="/A;/B;/C" <otherargs>
    ```

    CMAKE_PREFIX_PATH may also be set in the environment with paths separated by the appropriate element separator for the platform (":" on UNIX, ";" on Windows).

- GEANT4_USE_SYSTEM_CLHEP_GRANULAR (DEFAULT : OFF)
  - If set to ON, configure Geant4 to search for and use the Evaluator, Geometry, Random and Vector component libraries of CLHEP rather than the single CLHEP library.

    WARNING: This option should only be used if your project is locked into using the CLHEP granular libraries by other requirements. Use of the single CLHEP library is no different and simplifies the configuration and use of Geant4.

Options for Changing the Compiler and Build Flags

CMake will, by default, select the first C and C++ compilers it finds in your PATH. To specify the C and C++ compilers to be used, you can set the CC and CXX variables:

```
$ # ... assuming clang/clang++ are in the PATH ...

$ CC=clang CXX=clang++ cmake <otherargs>

$ # ... or ...

$ export CC=clang
$ export CXX=clang++
$ cmake <otherargs>
```

You can also use a full path should the compilers not be in the PATH or via the CMAKE_<LANG>_COMPILER options:

```
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ <otherargs>
```

Use of CMAKE_<LANG>_COMPILER will take precedence over any setting of CC or CXX in the environment or on the command line.

Whilst you *can* change the compiler after an initial configuration with CMake, *it is not recommended as you may need to reset some variables by hand*. If you are building Geant4 using several compilers and/or versions, we strongly recommend creating one build directory per compiler system. Whilst this takes extra disk space, it provides a clean separation between different builds and also allows fast incremental builds against a single source directory.

Geant4's CMake scripts configure a set of flags for use with each supported compiler as follows. CMAKE_CXX_FLAGS are always applied, with the CMAKE_CXX_FLAGS_<MODE> being appended when building in the given <MODE> (e.g. "Release"). If you are using an unsupported or unrecognised compiler, CMake will default to a standard and very simple set of flags.

- GNU Compiler Collection
  - CMAKE_CXX_FLAGS : -W -Wall -pedantic -Wno-non-virtual-dtor -Wno-long-long -Wwrite-strings -Wpointer-arith -Woverloaded-virtual -Wno-variadic-macros -Wshadow -pipe
  - CMAKE_CXX_FLAGS_RELEASE : -O3 -DNDEBUG -fno-trapping-math -ftree-vectorize -fno-math-errno
  - CMAKE_CXX_FLAGS_DEBUG : -g
  - CMAKE_CXX_FLAGS_RELWITHDEBINFO : -O2 -g
- Clang
  - CMAKE_CXX_FLAGS : -W -Wall -pedantic -Wno-non-virtual-dtor -Wno-long-long -Wwrite-strings -Wpointer-arith -Woverloaded-virtual -Wno-variadic-macros -Wshadow -pipe -Qunused-arguments

  - CMAKE_CXX_FLAGS_RELEASE : -O3 -DNDEBUG -fno-trapping-math -ftree-vectorize -fno-math-errno
  - CMAKE_CXX_FLAGS_DEBUG:-g
  - CMAKE_CXX_FLAGS_RELWITHDEBINFO:-O2 -g

- Microsoft Visual C++
  - CMAKE_CXX_FLAGS : -GR -EHsc -Zm200 -nologo -D_CONSOLE -D_WIN32 -DWIN32 -DOS -DXPNET -D_CRT_SECURE_NO_DEPRECATE
  - CMAKE_CXX_FLAGS_RELEASE:-MD -Ox -DNDEBUG
  - CMAKE_CXX_FLAGS_DEBUG:-MDd -Od -Zi
  - CMAKE_CXX_FLAGS_RELWITHDEBINFO:-MD -O2 -Zi

- Intel
  - CMAKE_CXX_FLAGS : -w1 -Wno-non-virtual-dtor -Wpointer-arith -Wwrite-strings -fp-model precise
  - CMAKE_CXX_FLAGS_RELEASE:-O3 -DNDEBUG
  - CMAKE_CXX_FLAGS_DEBUG:-g
  - CMAKE_CXX_FLAGS_RELWITHDEBINFO:-O2 -g

For the GNU, Clang and Intel compilers, an additional flag selecting the C++ standard to compile against will be set. By default, this will use the C++11 standard. This can be changed if the compiler version supports it by setting the GEANT4_BUILD_CXXSTD to the required standard, as described in *Advanced Options*.

When Geant4 is built with support for multithreading (GEANT4_BUILD_MULTITHREADED set to ON), the following additional flags are added to all build types for the GNU, Clang and Intel compilers:

- -ftls-model=initial-exec -pthread

Note that the model passed to the -ftls-model argument can be changed using the GEANT4_BUILD_TLS_MODEL option described in *Advanced Options*. The additional auto option to GEANT4_BUILD_TLS_MODEL does not set additional flags, leaving the selection of TLS model to the compiler.

Whilst we strongly recommend the default set of flags as described above, they may be modified. CMake is aware of the CFLAGS and CXXFLAGS variables, so these may be set on the command line or as environment variables like the CC and CXX options above. However, note that these will only *prepend* extra flags to the default CMAKE_<LANG>_FLAGS. If you need to completely change the compiler flags, then you can set CMAKE_<LANG>_FLAGS directly as a -D option to CMake. This will override all defaults set by Geant4's CMake scripts. Compiler flags can be interactively modified through the ccmake and CMake GUI interfaces. As compiler flags are an advanced option, you will need to activate viewing of advanced options. You may then edit the flags as you wish.

2.6.3 Postinstall Setup

If you chose the default installation paths, then your install of Geant4 is completely contained under the directory passed to `CMAKE_INSTALL_PREFIX`, with hierarchy

```
+- CMAKE_INSTALL_PREFIX
   +- bin/
   |  +- geant4-config    (UNIX ONLY)
   |  +- geant4.csh       (UNIX ONLY)
   |  +- geant4.sh        (UNIX ONLY)
   |  +- G4global.dll     (WINDOWS ONLY)
   |  +- ...
   +- include/
   |  +- Geant4/
   |     +- G4global.hh
   |     +- ...
   +- lib{64}/
   |  +- libG4global.{so,a,dylib,lib}
   |  +- ...
   |  +- Geant4-10.7.0/
   |     +- Geant4Config.cmake
   |     +- ...
   |     +- {Linux,Darwin}-{g++,clang}  (UNIX ONLY)
   +- share
      +- Geant4-10.7.0
         +- examples/
         +- data/         (IF GEANT4_INSTALL_DATA WAS SET)
         +- geant4make/
            +- geant4make.csh
            +- geant4make.sh
            +- config/
```

Required Environment Settings on UNIX

If you wish to make the Geant4 Toolkit programs and libraries available via your `PATH` and library path (`LD_LIBRARY_PATH` on Linux) together with default environment variables for locating datasets, you should source the relevant script in `CMAKE_INSTALL_PREFIX/bin`. Please note that `DYLD_LIBRARY_PATH` is not set on macOS as this variable is not propagated to programs due to SIP. This should not affect general running of Geant4 applications as the default macOS settings link and install the libraries with suitable install names and RPATHs.

On interactive Bourne shells (e.g. bash), do (assuming you are in `CMAKE_INSTALL_PREFIX/bin`):

```
$ source geant4.sh
```

This command can also be used to setup the environment for Geant4 in other Bourne shell scripts. You can also supply the full path to the script rather than changing to the directory containing it.

On interactive C shells, do (assuming you are in `CMAKE_INSTALL_PREFIX/bin`):

```
$ source geant4.csh
```

In an interactive session you can also supply the full path to the script rather than changing to the directory containing it. The C shell script cannot be sourced directly inside other shell scripts due to a limitation of the C shell which prevents the script being able to locate itself. If you need to source the C shell script inside another, then you can use the command:

```
$ cd CMAKE_INSTALL_PREFIX/bin ; source geant4.csh
```

where you should replace `CMAKE_INSTALL_PREFIX/bin` with the directory you installed `geant4.csh` in. You can also use the command:

```
$ source CMAKE_INSTALL_PREFIX/bin/geant4.csh CMAKE_INSTALL_PREFIX/bin
```

where as above you should replace `CMAKE_INSTALL_PREFIX/bin` with the directory where `geant4.csh` is located.

Required Environment Settings on Windows

On Windows, you should add the directory containing the Geant4 `.dll` files to your `PATH/Path` environment variable(s), and set the variables pointing to the datasets:

1. In *Search*, search for and select *System (Control Panel)*
2. Click the *Advanced system settings* link
3. Click the *Environment Variables* button
4. Select the `PATH` (or `Path`) entry in the *User variables* list, and click the *Edit* button. If `PATH` or `Path` are not present, click the *New* button and create one or the other.
5. In the popup *Edit User Variable* window, add the directory in which the Geant4 DLLs are installed to the Variable value entry of the `PATH` or `Path` variable (Note that on Windows, path entries are separated by semicolons). It's very important to keep any existing entries otherwise other programs may stop working correctly!
6. For each variable listed in *Environment Variables for Datasets*, create a new environment variable using the name and path listed there.
    - If you installed Geant4 with `GEANT4_INSTALL_DATA` set to `ON`, then the datasets will be present under `CMAKE_INSTALL_PREFIX/share/Geant4-10.7.0/data`.
    - Otherwise, set the paths to the locations you manually unpacked the datasets to.
7. Click *OK*, and then *OK* through remaining windows to close.

Environment Variables for Datasets

If you need to manually manage the datasets for the Geant4 data-driven physics models, zip/tarballs can be downloaded from the Geant4 distribution site. Simply unpack these to a location of your choice and set each environment variable listed below to the full path to the unpacked directory for the given dataset. You can use any method you like to set these variables.

Environment Variable	Value
G4ABLADATA	absolute path to the G4ABLA3.1 directory
G4ENSDFSTATEDATA	absolute path to the G4ENSDFSTATE2.3 directory
G4INCLDATA	absolute path to the G4INCL1.0 directory
G4LEDATA	absolute path to the G4EMLOW7.13 directory
G4LEVELGAMMADATA	absolute path to the PhotonEvaporation5.7 directory
G4NEUTRONHPDATA	absolute path to the G4NDL4.6 directory
G4PARTICLEXSDATA	absolute path to the G4PARTICLEXS3.1 directory
G4PIIDATA	absolute path to the G4PII1.3 directory
G4RADIOACTIVEDATA	absolute path to the RadioactiveDecay5.6 directory
G4REALSURFACEDATA	absolute path to the RealSurface2.2 directory
G4SAIDXSDATA	absolute path to the G4SAIDDATA2.0 directory

2.6.4 How to Use the Geant4 Toolkit Libraries

To build an application that uses the Geant4 Toolkit, it is necessary to include the Geant4 headers in the application C++ sources, and compile and link these to the Geant4 libraries. Full details on how to implement, build, and run a Geant4 application are provided in the Geant4 User's Guide for Application Developers.

Here we describe tools supplied with Geant4 to help with compilation and linking: a CMake `Geant4Config.cmake` config file and a UNIX-only command line program `geant4-config`. *A self-contained GNUmake system, "Geant4Make" is also supplied, but is deprecated since Geant4 10.0.*

CMake Build System: Geant4Config.cmake

The `Geant4Config.cmake` file installed by Geant4 is designed to be used with CMake's `find_package` command. When used, it sets several CMake variables and provides a mechanism for checking and activating optional features of Geant4. This allows you to use it in many ways in your CMake project to configure Geant4 for use by your application.

The most basic usage of `Geant4Config.cmake` in a CMakeLists.txt script is just to locate Geant4 with no requirements on its existence, version number or components

```
find_package(Geant4)
```

If Geant4 is an absolute requirement of the project, then you can use

```
find_package(Geant4 REQUIRED)
```

This will cause CMake to fail with an error should an install of Geant4 not be located. By default, CMake will look in several platform dependent locations for the `Geant4Config.cmake` file (see find_package for listings). If these are not sufficient to locate your install of Geant4, then the `Geant4_DIR` or `CMAKE_PREFIX_PATH` variables may be used. For example, if we have an install of Geant4 located in

```
+- opt/
   +- Geant4/
   +- lib/
      +- libG4global.so
      +- ...
      +- Geant4-10.7.0/
         +- Geant4Config.cmake
```

then we could pass the argument `-DGeant4_DIR=/opt/Geant4/lib/Geant4-10.7.0` (i.e. the directory holding `Geant4Config.cmake`) *or* `-DCMAKE_PREFIX_PATH=/opt/Geant4` to `cmake`.

When an install of Geant4 is found, the module sets a sequence of CMake variables that can be used elsewhere in the project:

- `Geant4_FOUND`

  Set to CMake boolean true if an install of Geant4 was found.

- `Geant4_INCLUDE_DIRS`

  Set to a list of directories containing Geant4 public headers.

- `Geant4_DEFINITIONS`

  The list of compile definitions needed to compile an application using Geant4. This is most typically used to correctly activate UI and Visualization drivers.

- `Geant4_LIBRARIES`

  Set to the list of CMake imported library targets that need to be linked to an application using Geant4. These targets provide CMake Usage Requirements so that linking to them ensures the correct set of include paths and definitions are applied to compile the dependent target.

- `Geant4_CXX_FLAGS`

  The compiler flags used to build this install of Geant4. Usually most important on Windows platforms.

- `Geant4_CXX_FLAGS_<CONFIG>`

  The compiler flags recommended for compiling Geant4 and applications in mode `CONFIG` (e.g. Release, Debug, etc). Usually most important on Windows platforms.

- `Geant4_CXXSTD`

  The C++ standard, e.g. "`c++11`" against which this install of Geant4 was compiled.

- `Geant4_TLS_MODEL`

  The thread-local storage model, e.g. "`initial-exec`" against which this install of Geant4 was compiled. Only set if the install was compiled with multithreading support.

- `Geant4_USE_FILE`

  A CMake script which can be included to handle certain CMake steps automatically. Most useful for very basic applications.

- `Geant4_builtin_clhep_FOUND`

  A CMake boolean which is set to true if this install of Geant4 was built using the internal CLHEP.

- `Geant4_system_clhep_ISGRANULAR`

  A CMake boolean which is set to true if this install of Geant4 was built using the system CLHEP and linked to the granular CLHEP libraries.

- `Geant4_builtin_expat_FOUND`

  A CMake boolean which is set to true if this install of Geant4 was built using the internal Expat.

- `Geant4_builtin_zlib_FOUND`

  A CMake boolean which is set to true if this install of Geant4 was built using the internal zlib.

- `Geant4_DATASETS`

  A CMake list of the names of the physics datasets used by physics models in Geant4. It is provided to help iterate over the `Geant4_DATASET_XXX_YYY` variables documented below.

- `Geant4_DATASET_<NAME>_ENVVAR`

  The name of the environment variable used by Geant4 to locate the dataset with name `<NAME>`.

- `Geant4_DATASET_<NAME>_PATH`

  The absolute path to the dataset with name `<NAME>`. Note that the setting of this variable does not guarantee the existence of the dataset, and no checking of the path is performed. This checking is not provided because the action you take on non-existing data will be application dependent.

  You can access the `Geant4_DATASET_XXX_YYY` variables in a CMake script in the following way:

```
find_package(Geant4_REQUIRED)                    # Find Geant4

foreach(dsname ${Geant4_DATASETS})               # Iterate over dataset names
  if(NOT EXISTS ${Geant4_DATASET_${dsname}_PATH})  # Check existence
    message(WARNING "${dsname} not located at ${Geant4_DATASET_${dsname}_PATH}")
  endif()
endforeach()
```

  A typical use case for these variables is to automatically set the dataset environment variables for your application without the use of the shell scripts described in *Postinstall Setup*. This could typically be via a shell script wrapper around your application, or runtime configuration of the application environment via the relevant C/C++ API for your system.

The typical usage of `find_package` and these variables to configure a build requiring Geant4 is thus:

```
find_package(Geant4 REQUIRED)                          # Find Geant4
set(CMAKE_CXX_FLAGS ${Geant4_CXX_FLAGS})               # Optional

add_executable(myg4app myg4app.cc)                     # Compile application
target_link_libraries(myg4app ${Geant4_LIBRARIES})     # Link it to Geant4
```

Alternatively, the CMake script pointed to by `Geant4_USE_FILE` may be included:

```
find_package(Geant4 REQUIRED)                          # Find Geant4
include(${Geant4_USE_FILE})                            # Auto configure includes/flags

add_executable(myg4app myg4app.cc)                     # Compile application
target_link_libraries(myg4app ${Geant4_LIBRARIES})     # Link it to Geant4
```

When included, the `Geant4_USE_FILE` script performs the following actions:

1. Adds the definitions in `Geant4_DEFINITIONS` to the global compile definitions via `add_definitions`
2. Appends the directories listed in `Geant4_INCLUDE_DIRS` to those the compiler uses for search for include paths, marking them as system include directories via `include_directories`
3. Prepends `Geant4_CXX_FLAGS` to `CMAKE_CXX_FLAGS`, and similarly for the extra compiler flags for each build mode (Release, Debug etc).

This use file is only intended for the most basic applications and is provided for backward compatibility. Projects should prefer linking directly to the Geant4 imported targets for ease of use and stronger linking guarantees.

A version number may be supplied to search for a Geant4 install *greater than or equal to* the supplied version, e.g.

```
find_package(Geant4 10.0 REQUIRED)
```

would make CMake search for a Geant4 install whose version number is greater than or equal to 10.0. An exact version number may also be specified:

```
find_package(Geant4 10.4.0 EXACT REQUIRED)
```

In both cases, CMake will fail with an error if a Geant4 install meeting these version requirements is not located.

Geant4 can be installed with many optional components, and the presence of these can also be required by passing extra "component" arguments. For example, to require that Geant4 is found *and* that it provides the Qt UI and visualization drivers, we can do

```
find_package(Geant4 REQUIRED qt)
```

In this case, if CMake finds a Geant4 install that does *not* support Qt, it will fail with an error. Multiple component arguments can be supplied, for example

```
find_package(Geant4 REQUIRED qt gdml)
```

requires that we find a Geant4 install that supports both Qt and GDML. If the components are found, the needed dependencies are refound and added as usage requirements on the Geant4 imported target libraries.

If you want to activate options only if they exist, you can use the pattern

```
find_package(Geant4 REQUIRED)
find_package(Geant4 QUIET OPTIONAL_COMPONENTS qt)
```

which will require CMake to locate a core install of Geant4, and then check for and activate Qt support if the install provides it, continuing without error otherwise. A key thing to note here is that you can call `find_package` multiple times to append configuration of components. If you use this pattern and need to check if a component was found, you can use the `Geant4_<COMPONENTNAME>_FOUND` variables described earlier to check the support.

The components which can be supplied to `find_package` for Geant4 are as follows:

- `static`

  `Geant4_static_FOUND` is `TRUE` if the install of Geant4 provides static libraries.

  Use of this component forces the variable `Geant4_LIBRARIES` to contain static library targets, if they are available. It can therefore be used to force static linking if your application requires this, but note that this does not guarantee that static versions of third party libraries will be used.

- `multithreaded`

  `Geant4_multithreaded_FOUND` is `TRUE` if the install of Geant4 was built with multithreading support.

  Note that this only indicates availability of multithreading support Multithreading in your applications requires creation and usage of the appropriate C++ objects and interfaces as described in the Application Developers Guide.

- `usolids`

  `Geant4_usolids_FOUND` is `TRUE` if the install of Geant4 was built with VecGeom replacing the Geant4 solids.

  Note that this only indicates that the replacement of Geant4 solids with VecGeom has taken place. Further information on the use of VecGeom applications is provided in the Application Developers Guide.

- `smartstack`

  `Geant4_smartstack_FOUND` is `TRUE` if the install of Geant4 was built with `G4SmartStack` use in `G4TrackingManager`.

- `php_as_hp`

  `Geant4_php_as_hp_FOUND` is `TRUE` if the install of Geant4 was built with ParticleHP as HP.

- `gdml`

  `Geant4_gdml_FOUND` is `TRUE` if the install of Geant4 was built with GDML support.

- `g3tog4`

  `Geant4_g3tog4_FOUND` is `TRUE` if the install of Geant4 provides the G3ToG4 library. If so, the G3ToG4 library is added to `Geant4_LIBRARIES`.

- `freetype`

  `Geant4_freetype_FOUND` is `TRUE` if the install of Geant4 was built with Freetype support.

- `hdf5`

  `Geant4_hdf5_FOUND` is `TRUE` if the install of Geant4 was built with HDF5 support.

- `ui_tcsh`

  `Geant4_ui_tcsh_FOUND` is `TRUE` if the install of Geant4 provides the TCsh command line User Interface. Using this component allows use of the TCsh command line interface in the linked application.

- `ui_win32`

  `Geant4_ui_win32_FOUND` is `TRUE` if the install of Geant4 provides the Win32 command line User Interface. Using this component allows use of the Win32 command line interface in the linked application.

- `motif`

  `Geant4_motif_FOUND` is `TRUE` if the install of Geant4 provides the Motif(Xm) User Interface and Visualization driver. Using this component allows use of the Motif User Interface and Visualization Driver in the linked application.

- `qt`

  `Geant4_qt_FOUND` is `TRUE` if the install of Geant4 provides the Qt User Interface and Visualization driver. Using this component allows use of the Qt User Interface and Visualization Driver in the linked application.

- `vis_dawn_network`

  `Geant4_vis_dawn_network_FOUND` is `TRUE` if the install of Geant4 provides the Client/Server network interface to DAWN visualization. Using this component allows use of the Client/Server DAWN Visualization Driver in the linked application.

- `vis_vrml_network`

  `Geant4_vis_vrml_network_FOUND` is `TRUE` if the install of Geant4 provides the Client/Server network interface to VRML visualization. Using this component allows use of the Client/Server VRML Visualization Driver in the linked application.

- `vis_raytracer_x11`

  `Geant4_vis_raytracer_x11_FOUND` is `TRUE` if the install of Geant4 provides the X11 interface to the RayTracer Visualization driver. Using this component allows use of the RayTracer X11 Visualization Driver in the linked application.

- `vis_opengl_x11`

  `Geant4_vis_opengl_x11_FOUND` is `TRUE` if the install of Geant4 provides the X11 interface to the OpenGL Visualization driver. Using this component allows use of the X11 OpenGL Visualization Driver in the linked application.

- `vis_opengl_win32`

`Geant4_vis_opengl_win32_FOUND` is `TRUE` if the install of Geant4 provides the Win32 interface to the OpenGL Visualization driver. Using this component allows use of the Win32 OpenGL Visualization Driver in the linked application.

- `vis_openinventor`

  `Geant4_vis_openinventor_FOUND` is `TRUE` if the install of Geant4 provides the OpenInventor Visualization driver. Using this component allows use of the OpenInventor Visualization Driver in the linked application.

- `ui_all`

  Activates all available UI drivers. Does not set any variables, and never causes CMake to fail.

- `vis_all`

  Activates all available Visualization drivers. Does not set any variables, and never causes CMake to fail.

Please note that whilst the above aims to give a complete summary of the functionality of `Geant4Config.cmake`, it only gives a sampling of the ways in which you may use it, and other CMake functionality, to configure your application. We also welcome feedback, suggestions for improvement and bug reports on `Geant4Config.cmake`.

Going further with CMake

The preceding sections show the minimal CMake scripting required to configure, build and install an application linking against the Geant4 libraries. If your project requires more advanced configuration, CMake provides tools such as compiler/platform identification and location/use of additional libraries/executables to link to/use. As this document is specific to Geant4, we do not cover more advanced usage of CMake and recommend that you consult the online manuals and tutorials supplied by Kitware together with the CMake Discourse Forum.

We also recommend the HEP Software Foundation CMake Tutorial and the CLIUtils Modern CMake Tutorial.

Other Unix Build Systems: `geant4-config`

If you wish to write your own Makefiles or use a completely different buildsystem for your application, a simple command line program named `geant4-config` is installed on Unix systems to help you query a Geant4 installation for locations and features. It is installed at:

```
+- CMAKE_INSTALL_PREFIX
   +- bin/
      +- geant4-config
```

It may be run using either a full or relative path, or directly if `CMAKE_INSTALL_PREFIX/bin` is in your `PATH`.

This program provides the following command line interface for querying various parameters of the Geant4 installation:

```
$ ./geant4-config --help
Usage: geant4-config [OPTION...]
  --prefix                output installation prefix of Geant4
  --version               output version for Geant4
  --cxxstd                C++ Standard compiled against
  --tls-model             Thread Local Storage model used
  --libs                  output all linker flags
  --cflags                output all preprocessor
                          and compiler flags
```

(continues on next page)

(continued from previous page)

```
  --libs-without-gui      output linker flags without
                          GUI components
  --cflags-without-gui    output preprocessor and compiler
                          flags without GUI components

  --has-feature FEATURE   output yes if FEATURE is supported,
                          or no if not supported

  --datasets              output dataset name, environment variable
                          and path, with one line per dataset

  --check-datasets        output dataset name, installation status and
                          expected installation location, with one line
                          per dataset

  --install-datasets      download and install any missing datasets,
                          requires a network connection and for the dataset
                          path to be user writable

Known Features:
 staticlibs[no]
 multithreading[no]
 clhep[yes]
 expat[no]
 zlib[yes]
 gdml[no]
 usolids[no]
 freetype[no]
 hdf5[no]
 g3tog4[no]
 qt[no]
 motif[no]
 raytracer-x11[no]
 opengl-x11[no]
 openinventor[no]

Help options:
 -?, --help              show this help message
 --usage                 display brief usage message
```

You are completely free to organise your application sources as you wish and to use any buildsystem that can interface with the output of `geant4-config`.

The `--cflags` argument will print the required compile definitions and include paths (in `-I<path>` format) to use Geant4 to stdout. Note that default header search paths for the compiler Geant4 was built with are filtered out of the output of `--cflags`.

The `--libs` argument will print the libraries (in `-L<path> -lname1 ... -lnameN` format) required to link with Geant4 to stdout. Note that this may include libraries for third party packages and may not be reliable for static builds. By default, all the flags and Geant4 libraries needed to activate all installed UI and Visualization drivers are provided in these outputs, but you may use the `--without-gui` variants of these arguments to suppress this.

You may also check the availability of features supported by the install of Geant4 with the `--has-feature` argument. If the argument to `--has-feature` is known to Geant4 *and* enabled in the installation, `yes` will be printed to stdout, otherwise `no` will be printed.

The `--datasets` argument may be used to print out a table of dataset names, environment variables and paths. No checking of the existence of the paths is performed, as the action to take on a non-existing dataset will depend

on your use case. The table is printed with one row per dataset, with space separated columns for the dataset name, environment variable name and path. As with `Geant4Config.cmake`, this information is provided to help you configure your application environment to locate the Geant4 datasets without a preexisting setup, if your use case demands this.

The `--check-datasets` argument may be used to check whether the datasets are installed in the location expected (as set by the configuration of Geant4). A table is printed with one row per dataset, with space separated columns for the dataset name, installation status and expected path. If the expected path is found, the status column will contain `INSTALLED`, otherwise it will contain `NOTFOUND`. Note that this check only verifies the existence of the dataset path. It does not validate that the dataset files are all present nor that the relevant environment variables are set.

If you did not use the `GEANT4_INSTALL_DATA` option to install data when Geant4 itself was installed, you can use the `--install-datasets` argument to perform this task at a later time. Running `geant4-config` with this argument will download, unpack and install each dataset to the location expected by the Geant4 installation. These steps require a working network connection, the local dataset installation path to be writable by the user running `geant4-config` and the presence of the `curl`, `openssl` and `tar` programs. Note that no changes to the environment are made by the data installation, so you may need to update this using the relevant scripts documented in *Postinstall Setup*.

Due to the wide range of possible use cases, we do not provide an example of using `geant4-config` to build an application. However, it should not require more than appending the output of `--cflags` to your compiler flags and that of `--libs` to the list of libraries to link to. We welcome feedback, suggestions for improvement and bug reports on `geant4-config`.

2.6.5 Appendices

Installing Coin3D and SoXt/Qt Bindings

- For example, on Mac, installing coin (needs Boost) and SoQt:

```
git clone --recurse-submodules https://github.com/coin3d/coin coin
cmake -Hcoin -Bcoin_build -G "Unix Makefiles" \
-DCMAKE_INSTALL_PREFIX=<install-directory>/coin3d/coin_install \
-DCOIN_BUILD_DOCUMENTATION=OFF -DCOIN_BUILD_MAC_FRAMEWORK=OFF \
-DBoost_INCLUDE_DIR=<install-directory>/boost/boost_1_72_0
cd coin_build
make -j8
sudo make install

git clone --recurse-submodules https://github.com/coin3d/soqt soqt
cmake -Hsoqt -Bsoqt_build -G "Unix Makefiles" \
-DCMAKE_INSTALL_PREFIX=<install-directory>coin3d/soqt_install \
-DCOIN_BUILD_MAC_FRAMEWORK=OFF \
-DCMAKE_PREFIX_PATH="<install-directory>/coin3d/coin_install;<Qt-installed-
↪directory>/5.15.0/clang_64"
cd soqt_build
make -j8
sudo make install
```

  Similarly for SoXt.

- On Unix, you should be able to install the needed libraries for Open Inventor using your package manager. They are:
  - Coin version 3 or above - see https://repology.org/project/coin3d/versions
  - SoQt version 1.6 - see https://repology.org/project/soqt/versions

  - SoXt version 1.4 - see https://repology.org/project/soxt/versions

  In all cases you should install both the basic version and the development version of the packages, e.g:

  - Coin3 Coin3-devel SoQt SoQt-devel (names may differ slightly)

  If you are unable to install as packages, the libraries can also be downloaded and built from source. The source code and build instructions for the Coin3d libraries are available from https://coin3d.github.io/

- On Windows, the libraries you need are Coin and SoWin (version 1.4). There may be Windows installers available.

Status of this Document

Guide for the installation and configuration of the Geant4 toolkit.

- Rev 1.0: First sphinx version implemented for Geant4 Release 10.4, 8th Dec 2017
- Rev 2.0: Updates and fixes in documentatio for GEANT4 Release 10.4, 15th May 2018
- Rev 3.0: GEANT4 Release 10.5, 11th December 2018
- Rev 3.1: GEANT4 Updates and fixes - especially to search functionality, 5th March 2019
- Rev 4.0: GEANT4 Release 10.6, 6th December 2019
- Rev 5.0: GEANT4 Release 10.7, 4th December 2020